

Repository Manager 3

Table of Contents

Table of Contents	1
Download.....	22
Related links	23
Download Archives - Repository Manager 3	23
Nexus Repository Manager 3.22.0-02	24
Nexus Repository Manager 3.21.2-03	24
Nexus Repository Manager 3.21.1-01	24
Nexus Repository Manager 3.21.0-05	25
Nexus Repository Manager 3.20.1-01	25
Nexus Repository Manager 3.20.0-04	25
Nexus Repository Manager 3.19.1-01	25
Nexus Repository Manager 3.19.0-01	26
Nexus Repository Manager 3.18.1-01	26
Nexus Repository Manager 3.18.0-01	26
Nexus Repository Manager 3.17.0-01	27
Nexus Repository Manager 3.16.2-01	27
Nexus Repository Manager 3.16.1-02	27
Nexus Repository Manager 3.16.0-01	27
Nexus Repository Manager 3.15.2-01	28
Nexus Repository Manager 3.15.1-01	28
Nexus Repository Manager 3.15.0-01	28
Nexus Repository Manager 3.14.0-04	29
Nexus Repository Manager 3.13.0-01	29
Nexus Repository Manager 3.12.1-01	29
Nexus Repository Manager 3.12.0-01	30
Nexus Repository Manager 3.11.0-01	30
Nexus Repository Manager 3.10.0-04	30
Nexus Repository Manager 3.9.0-01	31
Nexus Repository Manager 3.8.0-02	31
Nexus Repository Manager 3.7.1-02	31
Nexus Repository Manager 3.7.0-04	31

Nexus Repository Manager 3.6.2-01	32
Nexus Repository Manager 3.6.1-02	32
Nexus Repository Manager 3.6.0-02	32
Nexus Repository Manager 3.5.2-01	33
Nexus Repository Manager 3.5.1-02	33
Nexus Repository Manager 3.5.0-02	33
Nexus Repository Manager 3.4.0-02	33
Nexus Repository Manager 3.3.2-02	34
Nexus Repository Manager 3.3.1-01	34
Nexus Repository Manager 3.3.0-01	34
Nexus Repository Manager 3.2.1-01	35
Nexus Repository Manager 3.2.0-01	35
Nexus Repository Manager 3.1.0-04	35
Nexus Repository Manager 3.0.2-02	36
Nexus Repository Manager 3.0.1-01	36
Nexus Repository Manager 3.0.0-03	37
Release Notes	37
Key Features.....	38
Latest Release Notes.....	38
2020 Release Notes.....	39
Repository Manager 3.22.0	39
Repository Manager 3.21.2	41
Repository Manager 3.21.1	41
Repository Manager 3.21.0	41
2019 Release Notes.....	44
Repository Manager 3.20.1	44
Repository Manager 3.20.0	45
Repository Manager 3.19.1	48
Repository Manager 3.19.0	49
Repository Manager 3.18.1	51
Repository Manager 3.18.0	52
Repository Manager 3.17.0	53
Repository Manager 3.16.2	55
Repository Manager 3.16.1	55
Repository Manager 3.16.0	56
Repository Manager 3.15.2	63

Repository Manager 3.15.1	63
Repository Manager 3.15.0	63
2018 Release Notes.....	68
Repository Manager 3.14.0	68
Repository Manager 3.13.0	72
Repository Manager 3.12.1	74
Repository Manager 3.12.0	75
Repository Manager 3.11.0	76
Repository Manager 3.10.0	78
Repository Manager 3.9.0	80
Repository Manager 3.8.0	82
2017 Release Notes.....	85
Repository Manager 3.7.1	85
Repository Manager 3.7.0	86
Repository Manager 3.6.2	87
Repository Manager 3.6.1	88
Repository Manager 3.6.0	90
Repository Manager 3.5.2	91
Repository Manager 3.5.1	92
Repository Manager 3.5.0	92
Repository Manager 3.4.0	95
Repository Manager 3.3.2	100
Repository Manager 3.3.1	101
Repository Manager 3.3.0	102
Repository Manager 3.2.1	107
2016 Release Notes.....	110
Repository Manager 3.2.0	110
Repository Manager 3.1.0	113
Repository Manager 3.0.2	117
Repository Manager 3.0.1	118
Repository Manager 3.0.0	120
Milestone Releases.....	121
Milestone 7 Release	122
Milestone 6 Release.....	124
Milestone 5 Release.....	125
Milestone 4 Release.....	127
Milestone 3 Release.....	129

Milestone 2 Release.....	131
Milestone 1 Release.....	133
System Requirements.....	137
Supported Versions	137
Host Operating System	137
Dedicated Operating System User Account.....	138
Adequate File Handle Limits	138
Linux	138
Mac OSX	139
Windows	141
Docker.....	141
Java	141
CPU	142
Memory.....	142
Configurable Memory Types.....	142
Memory Requirements	143
General Memory Guidelines	143
Instance Memory Sizing Profiles	143
Example Maximum Memory Configurations	144
Advanced Database Memory Tuning	145
Temporary Directory	145
Disk Space.....	145
File Systems to avoid.....	145
Web Browser	146
Upgrade Compatibility - Repository Manager 2 to 3.....	146
Requirement: Always Use Latest Versions	146
Requirement: Always Use Same License	146
Supported Nexus Repository Manager Upgrade Paths.....	147
Minimally Compatible Server Versions for Lifecycle Features	148
Repository Manager 2 to 3 Feature Equivalency	148
Repository Manager Feature Matrix	153

OBR	155
Replication	155
Custom Metadata	155
Plugins	155
Repository Manager Pro Features	156
Atlassian Crowd Support.....	156
Staging and Build Promotion	156
Typical Staging Workflow.....	156
Tagging.....	157
User Token Support	157
High Availability	157
Typical HA Configuration	158
Repository Health Check	158
Example Health Check Overview	158
Customer Success.....	159
Enterprise Support	159
Group Blob Stores.....	159
Example Group Blob Store	159
SAML Authentication and Single Sign-On	160
Repository Manager Concepts.....	160
Topics in this section:.....	160
Components, Repositories, and Repository Formats	161
Components.....	161
Assets.....	162
Components in Repositories.....	162
An Example - Maven Repository Format	163
The Central Repository	163
Component Coordinates and the Repository Format.....	164
Release and Snapshot Repositories	166
Managing Repositories.....	167
Capabilities of a Repository Manager	167

Advantages of Using a Repository Manager.....	168
Software Supply Chain Automation.....	168
Installation.....	169
Topics in the section:.....	169
Java Runtime Environment	169
Installation Methods	170
Installing and Running with the Distribution Archive.....	170
Installing with Docker	171
Run as a Service.....	172
Linux	172
Windows	174
Mac OS X	175
Run Behind a Reverse Proxy.....	176
Example: Reverse Proxy on Restricted Ports	177
Example: Reverse Proxy Virtual Host at Custom Context Path	178
Example: Reverse Proxy SSL Termination at Base Path	180
Apache httpd with npm Repositories	181
Accessing the User Interface	182
Directories	183
Installation Directory.....	183
Data Directory.....	184
Configuring the Runtime Environment	185
Configuring Memory	187
Configuring Java Tool Options	188
Changing the HTTP Port.....	188
Changing the Context Path	189
Configuring the Data Directory.....	189
Configuring the Temporary Directory	189
Post Install Checklist	190
Step 1: Change the admin password.....	190
Step 2: Configure Anonymous Access	191
Step 3: Change the Administrative Email Address	191
Step 4: Configure the SMTP Settings	191
Step 5: Configure Default HTTP and HTTPS Proxy Settings	192

Step 6: Setup a Backup procedure for your server	192
Securing Nexus Repository Manager	192
Limit IPs that can be reached from your NXRM host.....	192
Privileges and service account	192
Containerization.....	192
Upgrading	193
Why Upgrade to Nexus Repository Manager 3?	193
Upgrading from 3.x to 3.y	194
Upgrading a Standalone Instance.....	194
Upgrading a High Availability Cluster (HA-C).....	194
Upgrading from 2.x to 2.y	194
Upgrading from 2.x to 3.y	195
Upgrade Process and Expectations.....	196
What Is Upgraded	196
What Is Not Upgraded	197
Repository Format Support	197
Designing Your Upgrade Plan	198
Supported Installation Scenarios.....	198
Data Transfer Methods.....	199
HTTP Downloading	199
File System Copying	199
File System Hard Linking.....	199
File System Considerations.....	200
Upgrade Details for Specific Elements	200
Repository IDs	200
Repository Groups	200
User Tokens	201
Repository Health Check and SSL Health Check.....	201
NuGet API Key	201
HTTP(S) Proxy Configuration.....	201
Licensing	201
IQ Server	202
Staging.....	202
Security Compatibility.....	202

Optimization, Performance, and Tuning	202
Upgrade Procedures	203
Starting the Upgrade.....	204
Enabling the Upgrade Capability in Version 2.x	204
Enabling the Upgrade Capability in Version 3.x	204
Upgrading Content.....	205
Running the Upgrade	206
After the Upgrade.....	207
Restarting an Upgrade.....	208
User Interface.....	209
Topics in this section:.....	209
User Interface Overview	209
Searching for Components	213
Search Criteria and Component Attributes	214
Keyword search query string handling	217
Search Results	218
Preconfigured Searches	219
Example Use Case - SHA-1 Search	221
Viewing Component Information.....	221
Viewing Asset Information.....	223
Browsing Repositories and Repository Groups	227
Working with Your User Profile	229
Changing Your Password	229
Uploading Components.....	230
Before you begin	230
How to upload a component.....	231
Advanced usage.....	232
Viewing Tags.....	232
Configuration.....	233
Topics in this section:.....	233
Administration Menu	234
Repository Management	234
Blob Stores	235

Proxy Repository	238
Hosted Repository	239
Repository Group	240
Managing Repositories and Repository Groups	240
Repository Management Example.....	249
Content Selectors	251
Support Features	254
Logging and Log Viewer.....	255
Metrics.....	256
Support ZIP.....	257
System Information	258
System Configuration	258
Accessing Information About Bundles.....	259
Accessing and Configuring Capabilities	259
Email/SMTP Server Configuration.....	261
Nodes.....	263
Base URL Creation	263
UI Settings	264
HTTP and HTTPS Request and Proxy Settings.....	266
Configuring and Executing Tasks	269
License Management	280
Uploading a License	280
Monitoring Recent Connections for Product License Compliance	280
Storage Guide.....	281
Introduction	281
Terminology	281
Fields	282
Planning.....	282
Auditing	286
Routing Rules	287
Creating or Modifying a Routing Rule	287
Testing Routing Rules.....	288
Deleting a Routing Rule	289
Assigning a Rule to a Repository	289
Retry Limit Configuration.....	289
Introduction	289

System Properties.....	289
RetryControllerBean.....	290
References	293
Backup and Restore.....	294
Topics in this section:.....	294
Prepare a Backup.....	294
Blob Store Backup.....	295
Database Backup	295
Configure and Run the Backup Task	295
Restore Exported Databases.....	296
Cleanup Policies	297
Summary	297
Cleanup Policies	298
Criteria	299
Cleanup Task.....	304
General Guide (FAQ)	304
What Cleanup Policies are built in?	304
Is there anything more I need to do other than apply policies to free up space?	304
How Docker Repositories Use Space	305
How can I tell what repositories are using the most space?	306
How do Cleanup Policies impact other "delete" tasks?.....	306
What is the needed security to use this feature?.....	307
What if I'm on an older version of NXRM3?.....	307
Any other tips?	307
Keeping Disk Usage Low	307
Tasks	308
For More Complex Cleanup Scenarios	308
Examining Blobstore Space Usage.....	308
Totally Out of Space / Seeing Errors?.....	309
Security	309
Topics in this section:.....	310
Realms	310

Privileges	313
Privilege Types	314
Privilege Actions	315
Example: Creating an Application Privilege	316
Example: Creating a Repository View Privilege	316
Roles	317
Mapping External Groups to Nexus Roles.....	320
Users	320
Anonymous Access	323
LDAP	324
Enabling the LDAP Authentication Realm	324
LDAP Connection and Authentication	325
User and Group Mapping.....	327
SAML	331
NXRM3 SAML Support	331
Integrating NXRM3 with an Identity Provider.....	331
SAML Configuration Details.....	333
User SAML Workflow.....	334
Revoking User Access	335
Security Setup with User Tokens	335
Enabling and Resetting User Tokens.....	336
Accessing User Tokens in Realms	336
Accessing and Using Your User Token	336
Authentication via Remote User Token	337
Configuring SSL	338
Outbound SSL - Trusting SSL Certificates of Remote Repositories	339
Outbound SSL - Trusting SSL Certificates Globally	340
Outbound SSL - Trusting SSL Certificates Using Keytool.....	343
Inbound SSL - Configuring to Serve Content via HTTPS	344
Default Role	346
Atlassian Crowd Support.....	347
Topics in this section:.....	347
Preparing Atlassian Crowd.....	348
Compatibility	348

Adding a New Application to the Crowd Server	348
Configure Crowd Integration	349
Enable the Crowd Capability	349
Configure Pro to Trust Crowd's Secure URL (Optional)	350
Configure Nexus Repository Manager Pro Crowd Security	351
Staging.....	352
Overview	352
Building Blocks.....	352
Workflow	353
REST Endpoints.....	354
Promotion.....	354
Deletion.....	355
Integrations	356
NXRM 2 Migration.....	356
Limitations.....	356
Tagging.....	357
Overview	357
Endpoints.....	357
Add Tag	357
List Tags	358
Get Tag	359
Update Tag	360
Associate Components with a Tag.....	360
Tagging During Component Upload	361
Disassociate Components from a Tag.....	362
Privileges	362
Cleanup Task.....	363
Best Practices	364
Formats	364
Apt Repositories.....	367
Introduction	367
Proxying Apt Repositories	367

Hosting Apt Repositories	368
Deploying Packages to Apt Hosted Repositories	369
Browsing Apt Repositories and Searching Packages	369
Configuring Apt Client	369
Taking a Snapshot of a repository metadata.....	370
Bower Repositories.....	371
Introduction	371
Proxying Bower Repositories	372
Hosting Bower Repositories.....	372
Bower Repository Groups.....	372
Installing Bower	373
Configuring Bower Package Download	373
Browsing Bower Repositories and Searching Packages.....	375
Registering Bower Packages	375
CocoaPods Repositories	376
Introduction	376
Support information.....	376
Proxying CocoaPods Repositories	377
Configure CocoaPods client	377
Conan Repositories.....	378
Introduction	378
Conan client setup	379
Proxying Conan Repositories	379
Conan client configuration to work with NXRM	380
Conda Repositories.....	381
Introduction	381
Supported features	381
Repository configuration	382
Configuring the Conda client to use NXRM.....	382
Docker Registry	383
Topics in this section:.....	383
SSL and Repository Connector Configuration	384
Proxy Repository for Docker	386
Hosted Repository for Docker (Private Registry for Docker)	387
Grouping Docker Repositories	388
Content Selectors and Docker	388
Authentication.....	394

Accessing Repositories.....	395
Searching.....	396
Pulling Images.....	397
Pushing Images	398
Scaling Repositories	399
Foreign Layers.....	400
Docker Content Trust.....	402
Git LFS Repositories	403
Introduction	403
Requirements	404
Creating a Hosted Git LFS Repository	404
Installing Git LFS Locally	404
Configuring Git LFS Locally	405
Go Repositories.....	406
Introduction	406
Proxying Go Repositories	407
Grouping Go Repositories	407
Go configuration examples	408
Helm Repositories	409
Introduction	409
Proxying Helm Repositories.....	409
Hosting Helm Repositories	409
Uploading Charts to Helm Hosted Repositories	410
Uploading Provenance Files to Helm Hosted Repositories	410
Configuring Helm Client	410
Maven Repositories	411
Introduction	411
Maven Repository Format	412
Proxying Maven Repositories.....	413
Hosting Maven Repositories	414
Grouping Maven Repositories.....	414
Browsing and Searching Maven Repositories	414
Configuring Apache Maven	415
Configuring Apache Ant and Apache Ivy	418
Configuring Apache Ant and Eclipse Aether	418
Configuring Gradle	419
SBT.....	420

Leiningen	421
npm Registry	422
Introduction	422
Proxying npm Registries.....	423
Private npm Registries.....	423
Grouping npm Registries	423
Browsing npm Registries and Searching Modules	424
Configuring npm	424
npm Security	425
Publishing npm Packages	427
Deprecating npm Packages	427
NuGet Repositories	428
NuGet Repository Format.....	428
NuGet Proxy Repositories	429
NuGet Hosted Repositories.....	432
Grouping NuGet Repositories	433
Accessing Packages in Repositories and Groups	434
Deploying Packages to NuGet Hosted Repositories	434
Integration with Visual Studio	435
p2 Repositories	436
Introduction	436
Supported features	437
Proxying p2 Repositories.....	437
Browsing p2 Repositories and Searching Packages	437
Configuring the Eclipse IDE to use NXRM	437
PyPI Repositories	440
Introduction	440
Proxying PyPI Repositories	441
Hosting PyPI Repositories.....	441
Grouping PyPI Repositories	442
Installing PyPI Client Tools.....	442
Configuring PyPI Client Tools	442
SSL Usage for PyPI Repositories	444
Browsing PyPI Repositories and Searching Packages.....	445
Uploading PyPI Packages	445
R Repositories	445
Introduction	446

Proxying R Repositories	446
Hosting R Repositories.....	446
Deploying Packages to R Hosted Repositories.....	447
R Repository Groups	447
Configuring R Client	448
Raw Repositories	448
Introduction	449
Creating a Hosted Raw Repository	449
Creating and Deploying a Maven Site.....	449
Proxying and Grouping Raw Repositories	453
Uploading Files to Hosted Raw Repositories.....	454
RubyGems Repositories	454
Introduction	455
Proxying RubyGem Repositories.....	456
Private Hosted RubyGem Repositories	456
Grouping RubyGem Repositories.....	456
Using RubyGem Repositories.....	457
Pushing Gems	459
Yum Repositories.....	460
Introduction	460
Proxying Yum Repositories	460
Hosting Yum Repositories.....	461
Grouping Yum Repositories	461
Deploying Packages to Yum Hosted Repositories	462
Comps.xml (package grouping).....	463
Installing Yum	463
Configuring Yum Client.....	463
Browsing Yum Repositories and Searching Packages.....	464
REST and Integration API	464
REST API.....	464
Topics in this section:.....	465
Scripting API.....	465
Assets API	466
Introduction	466
Endpoints.....	467

Blob Store API	468
Introduction	469
Endpoints.....	469
Components API.....	469
Introduction	469
Endpoints.....	469
Email API	477
Introduction	477
Endpoints.....	477
IQ Server API	477
Introduction	477
Endpoints.....	478
Licensing API	478
Introduction	478
Endpoints.....	478
Lifecycle API.....	478
Introduction	479
Endpoints.....	479
Maintenance API.....	480
Introduction	480
Endpoints.....	480
Nodes API.....	484
Introduction	484
Endpoints.....	484
Read-Only API.....	485
Introduction	486
Endpoints.....	486
Repositories API.....	488
Introduction	488
Endpoints.....	488
Search API.....	489
Introduction	489
Endpoints.....	489
Security Management API.....	496
Content Selectors	496

LDAP	496
Privileges	497
Roles	497
Users	497
User Sources	497
Script API.....	497
Introduction	498
Endpoints.....	498
Writing Scripts.....	502
Managing and Running Scripts	504
Examples	506
Status API.....	509
Introduction	509
Endpoints.....	509
Support API	510
Introduction	510
Endpoints.....	510
Tasks API	511
Introduction	511
Endpoints.....	511
Pagination	514
Webhooks.....	515
Using Webhooks	515
Topics in this section:.....	516
Enabling A Global Webhook Capability	516
Enabling A Repository Webhook Capability	518
Working With HMAC Payloads.....	519
Example Headers And Payloads.....	520
Example Audit Payload.....	521
Example Repository Payload.....	522
Example Repository Asset Payload	522
Example Repository Component Payload	523
Bundle Development.....	524

Topics in this section:	525
Installing Bundles	525
Installing Bundles Into 3.15.0 and Later	526
Installing Bundles Into 3.14.0 and Earlier	526
Bundle Development Overview	527
Inheriting from the nexus-plugins Parent	527
Support for a New Repository Format	528
Format, Recipe, and Facet	529
Storage	530
User Interface	530
Tasks	530
Contributing Bundles	530
High Availability	531
Overview	531
Software and Hardware Requirements	532
Known Issues and Limitations	532
Topics in this section	532
Configuring Nodes	533
What is a Node?	533
How many Nodes will you need?	533
What data is replicated between Nodes?	534
Configuring Blob Stores	534
What is a Blob Store?	534
Choosing a Type for your Blob Stores	535
Determining a Backup Solution	538
Configuring Hazelcast	538
Network Preparation	538
Node Discovery	539
Designing your Cluster Backup/Restore Process	543
High-level Overview	543
Backing up Shared File Systems for your Blob Store(s)	544
Backing up local storage for your Nodes	544
Restoring from backup	545

Initial Setup - High Availability.....	546
Node Deployment Steps for High Availability	547
Enabling High Availability on an existing Nexus Repository Manager Deployment	549
Operating your cluster	550
Startup and Confirming Node Connectivity.....	550
Shutdown.....	551
Backup	551
Node Names	552
Maintaining log files	552
Verifying Synchronization.....	553
Monitoring Node Health	555
Task Management	555
JMX Lifecycle Operations	556
JMX Maintenance Operations	558
Upgrading your cluster	563
Load Balancing	564
Overview	564
Ports	565
Sticky Sessions	565
Monitoring Node Health	565
SSL Termination.....	566
sitemap.xml.....	566
Quick Start Guide - Proxying Maven and NPM.....	566
Requirements	566
Part 1 - Installing and Starting Nexus Repository Manager 3	567
Part 2 - Proxying Maven and npm Components	567
Maven	567
npm	569
Part 3 - Viewing Proxied Components	570

Welcome to the Help Portal for Nexus Repository Manager (NXRM) 3.

If you are developing software without a repository manager you are likely missing a number of opportunities to reduce some pretty obvious inefficiencies. If everyone on your team has to hit public repositories like the Central Repository to download components, you are missing out on some simple gains in speed and efficiency. If you don't have a local place to deploy components you are forced to share binary components using half-measures and compromises such as storing binaries in source control. Stop developing in the Dark Ages, read this guide, and start using a repository manager. Trust us, once you start using a Nexus Repository Manager, you'll wonder how you ever functioned without it.

For those new to Repository Manager, we've collected a number of topics perfect for getting introduced quickly and efficiently. In addition, you can always check out a variety of additional content available via [Sonatype Learning](#)¹.

- ⓘ All pages below and across the NXRM3 documentation reflect against the latest version of NXRM3 except where mentioned. You can get the latest NXRM3 via the [Download](#)(see page 22) page. If you have an older version, some of the details herein may not be accurate for you. We always encourage upgrading to receive the latest features and bug fixes!

Installation

- [Installation](#)(see page 169)

Configuration

- [Configuration](#)(see page 233)
- [Using the User Interface](#)(see page 209)

Upgrading

- [Upgrading Guide](#)(see page 193)
- [Upgrading 3.x to 3.y \(support article\)](#)²

Repository Formats

- [Apt Repositories](#)(see page 367)
- [Bower Repositories](#)(see page 371)
- [CocoaPods Repositories](#)(see page 376)
- [Conan Repositories](#)(see page 378)
- [Conda Repositories](#)(see page 381)
- [Docker Registry](#)(see page 383)
- [Git LFS Repositories](#)(see page 403)

¹ <https://help.sonatype.com/display/SL/Sonatype+Learning>

² <https://support.sonatype.com/hc/en-us/articles/217967608>

- [Go Repositories\(see page 406\)](#)
- [Helm Repositories\(see page 409\)](#)
- [Maven Repositories\(see page 411\)](#)
- [npm Registry\(see page 422\)](#)
- [NuGet Repositories\(see page 428\)](#)
- [p2 Repositories\(see page 436\)](#)
- [PyPI Repositories\(see page 440\)](#)
- [R Repositories\(see page 445\)](#)
- [Raw Repositories\(see page 448\)](#)
- [RubyGems Repositories\(see page 454\)](#)
- [Yum Repositories\(see page 460\)](#)

Scripting and Development

- [REST and Integration API\(see page 464\)](#)
- [Bundle Development\(see page 524\)](#)

To learn more about Nexus solutions and licenses, see the knowledge base article [Sonatype Product and Solutions Overview](#)³ or contact sales@sonatype.com⁴.

Download

Distributions for Nexus Repository Manager 3 are available for the 64-bit versions for Apple OSX, Microsoft Windows and Unix/Linux. They contain all necessary resources to install and run the repository manager. You can download the archive file for your operating system.

The download on this page is used for both Nexus Repository Manager PRO and Nexus Repository Manager OSS. See [License Management\(see page 280\)](#) for information on getting your OSS version to PRO with your professional license.

 On this page you can download the most recent version of Nexus Repository Manager. If you're looking for older versions, visit the [download archive\(see page 23\)](#). Also, you can access [earlier versions via Docker images](#)⁵ back to version 3.0.0.

Archive files are Gzip TAR (TGZ) or ZIP files and are used for installation without a graphical user interface, instead using command line-based interaction.

³ <https://support.sonatype.com/hc/en-us/articles/215716807>

⁴ <mailto:sales@sonatype.com>

⁵ <https://hub.docker.com/r/sonatype/nexus3/tags/>

After download is complete, consult [Installation\(see page 169\)](#) and [System Requirements\(see page 137\)](#) for further information about installation and necessary setup you may need. If you are upgrading an existing NXRM3 instance, see [the 3.x to 3.y subsection\(see page 194\)](#). If you are upgrading from NXRM2 to NXRM3, see [the 2.x to 3.y subsection\(see page 195\)](#) and further subpages in the Upgrading section.

Latest Version: 3.22.0-02

Operating System	Link for Download
Unix archive	https://download.sonatype.com/nexus/3/latest-unix.tar.gz (ASC⁶ , MD5⁷ , SHA1⁸)
Windows archive	https://download.sonatype.com/nexus/3/latest-win64.zip (ASC⁹ , MD5¹⁰ , SHA1¹¹)
OSX archive	https://download.sonatype.com/nexus/3/latest-mac.tgz (ASC¹² , MD5¹³ , SHA1¹⁴)

Related links

Software	Link for Download
Docker Image	https://hub.docker.com/r/sonatype/nexus3/
Cloud Templates	Cloud Deployments¹⁵
Jenkins Plugin	Nexus Platform Plugin for Jenkins¹⁶

Download Archives - Repository Manager 3

Docker images for Nexus Repository Manager are available through the [Docker Hub¹⁷](#). Information about templates can be found on the [Cloud Deployments¹⁸](#) page.

6 <https://download.sonatype.com/nexus/3/latest-unix.tar.gz.asc>

7 <https://download.sonatype.com/nexus/3/latest-unix.tar.gz.md5>

8 <https://download.sonatype.com/nexus/3/latest-unix.tar.gz.sha1>

9 <https://download.sonatype.com/nexus/3/latest-win64.zip.asc>

10 <https://download.sonatype.com/nexus/3/latest-win64.zip.md5>

11 <https://download.sonatype.com/nexus/3/latest-win64.zip.sha1>

12 <https://download.sonatype.com/nexus/3/latest-mac.tgz.asc>

13 <https://download.sonatype.com/nexus/3/latest-mac.tgz.md5>

14 <https://download.sonatype.com/nexus/3/latest-mac.tgz.sha1>

15 <https://help.sonatype.com/display/NXI/Cloud+Deployments>

16 <https://help.sonatype.com/display/NXI/Nexus+Platform+Plugin+for+Jenkins#NexusPlatformPluginforJenkins-RepositoryManager3Integration>

17 <https://hub.docker.com/r/sonatype/nexus3/>

18 <https://help.sonatype.com/display/NXIM/Cloud+Deployments>

Nexus Repository Manager 3.22.0-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.22.0-02-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.22.0-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.22.0-02-mac.tgz

 Nexus Repository Manager 3 is evolving rapidly and we discourage using older versions. However, they are below in case you have need.

Nexus Repository Manager 3.21.2-03

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.21.2-03-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.21.2-03-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.21.2-03-mac.tgz

Nexus Repository Manager 3.21.1-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.21.1-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.21.1-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.21.1-01-mac.tgz

Nexus Repository Manager 3.21.0-05

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.21.0-05-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.21.0-05-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.21.0-05-mac.tgz

Nexus Repository Manager 3.20.1-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.20.1-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.20.1-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.20.1-01-mac.tgz

Nexus Repository Manager 3.20.0-04

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.20.0-04-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.20.0-04-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.20.0-04-mac.tgz

Nexus Repository Manager 3.19.1-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.19.1-01-unix.tar.gz

Operating System	Link for Download
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.19.1-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.19.1-01-mac.tgz

Nexus Repository Manager 3.19.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.19.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.19.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.19.0-01-mac.tgz

Nexus Repository Manager 3.18.1-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.18.1-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.18.1-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.18.1-01-mac.tgz

Nexus Repository Manager 3.18.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.18.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.18.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.18.0-01-mac.tgz

Nexus Repository Manager 3.17.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.17.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.17.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.17.0-01-mac.tgz

Nexus Repository Manager 3.16.2-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.16.2-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.16.2-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.16.2-01-mac.tgz

Nexus Repository Manager 3.16.1-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.16.1-02-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.16.1-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.16.1-02-mac.tgz

Nexus Repository Manager 3.16.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.16.0-01-unix.tar.gz

Operating System	Link for Download
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.16.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.16.0-01-mac.tgz

Nexus Repository Manager 3.15.2-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.15.2-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.15.2-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.15.2-01-mac.tgz

Nexus Repository Manager 3.15.1-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.15.1-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.15.1-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.15.1-01-mac.tgz

Nexus Repository Manager 3.15.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.15.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.15.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.15.0-01-mac.tgz

❗ Known Security Issue

Versions of NXRM3 up to and including 3.14.0 are vulnerable to a critical security vulnerability. They are no longer recommended for use. For details, please see [CVE-2019-7238¹⁹](https://support.sonatype.com/hc/en-us/articles/360017310793-CVE-2019-7238-Nexus-Repository-Manager-3-Missing-Access-Controls-and-Remote-Code-Execution-February-5th-2019).

Nexus Repository Manager 3.14.0-04

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.14.0-04-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.14.0-04-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.14.0-04-mac.tgz

Nexus Repository Manager 3.13.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.13.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.13.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.13.0-01-mac.tgz

Nexus Repository Manager 3.12.1-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.12.1-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.12.1-01-win64.zip

¹⁹ <https://support.sonatype.com/hc/en-us/articles/360017310793-CVE-2019-7238-Nexus-Repository-Manager-3-Missing-Access-Controls-and-Remote-Code-Execution-February-5th-2019>

Operating System	Link for Download
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.12.1-01-mac.tgz

Nexus Repository Manager 3.12.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.12.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.12.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.12.0-01-mac.tgz

Nexus Repository Manager 3.11.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.11.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.11.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.11.0-01-mac.tgz

Nexus Repository Manager 3.10.0-04

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.10.0-04-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.10.0-04-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.10.0-04-mac.tgz

Nexus Repository Manager 3.9.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.9.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.9.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.9.0-01-mac.tgz

Nexus Repository Manager 3.8.0-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.8.0-02-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.8.0-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.8.0-02-mac.tgz

Nexus Repository Manager 3.7.1-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.7.1-02-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.7.1-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.7.1-02-mac.tgz

Nexus Repository Manager 3.7.0-04

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.7.0-04-unix.tar.gz

Operating System	Link for Download
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.7.0-04-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.7.0-04-mac.tgz

Nexus Repository Manager 3.6.2-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.6.2-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.6.2-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.6.2-01-mac.tgz

Nexus Repository Manager 3.6.1-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.6.1-02-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.6.1-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.6.1-02-mac.tgz

Nexus Repository Manager 3.6.0-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.6.0-02-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.6.0-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.6.0-02-mac.tgz

Nexus Repository Manager 3.5.2-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.5.2-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.5.2-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.5.2-01-mac.tgz

Nexus Repository Manager 3.5.1-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.5.1-02-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.5.1-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.5.1-02-mac.tgz

Nexus Repository Manager 3.5.0-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.5.0-02-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.5.0-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.5.0-02-mac.tgz

Nexus Repository Manager 3.4.0-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.4.0-02-unix.tar.gz

Operating System	Link for Download
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.4.0-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.4.0-02-mac.tgz

Nexus Repository Manager 3.3.2-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.3.2-02-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.3.2-02-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.3.2-02-mac.tgz

Nexus Repository Manager 3.3.1-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.3.1-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.3.1-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.3.1-01-mac.tgz

Nexus Repository Manager 3.3.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.3.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.3.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.3.0-01-mac.tgz

Nexus Repository Manager 3.2.1-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.2.1-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.2.1-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.2.1-01-mac.tgz

Nexus Repository Manager 3.2.0-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.2.0-01-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.2.0-01-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.2.0-01-mac.tgz

Nexus Repository Manager 3.1.0-04

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.1.0-04-unix.tar.gz
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.1.0-04-win64.zip
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.1.0-04-mac.tgz

Note: The [GUI installer distribution development is suspended²⁰](#) and is no longer an install option until further notice.

²⁰ <https://sonatype.zendesk.com/hc/en-us/articles/229935227>

i **Repository Manager 3.1.0** and newer can accept a Pro license installation and includes Pro features when activated.

Repository Manager 3.0.x does not require a license to be installed, and does not include Pro features, however all existing licensed repository manager accounts are entitled to full technical support.

Nexus Repository Manager 3.0.2-02

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.0.2-02-unix.tar.gz
Unix installer	http://download.sonatype.com/nexus/3/nexus-3.0.2-02-unix.sh
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.0.2-02-win64.zip
Windows installer	http://download.sonatype.com/nexus/3/nexus-3.0.2-02-win64.exe
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.0.2-02-mac.tgz
OSX installer	http://download.sonatype.com/nexus/3/nexus-3.0.2-02-mac.dmg

Nexus Repository Manager 3.0.1-01

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.0.1-01-unix.tar.gz
Unix installer	http://download.sonatype.com/nexus/3/nexus-3.0.1-01-unix.sh
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.0.1-01-win64.zip
Windows installer	http://download.sonatype.com/nexus/3/nexus-3.0.1-01-win64.exe

Operating System	Link for Download
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.0.1-01-mac.tgz
OSX installer	http://download.sonatype.com/nexus/3/nexus-3.0.1-01-mac.dmg

Nexus Repository Manager 3.0.0-03

Operating System	Link for Download
Unix archive	http://download.sonatype.com/nexus/3/nexus-3.0.0-03-unix.tar.gz
Unix installer	http://download.sonatype.com/nexus/3/nexus-3.0.0-03-unix.sh
Windows archive	http://download.sonatype.com/nexus/3/nexus-3.0.0-03-win64.zip
Windows installer	http://download.sonatype.com/nexus/3/nexus-3.0.0-03-win64.exe
OSX archive	http://download.sonatype.com/nexus/3/nexus-3.0.0-03-mac.tgz
OSX installer	http://download.sonatype.com/nexus/3/nexus-3.0.0-03-mac.dmg

Release Notes

We're always improving Nexus Repository Manager 3 products and features based on customer feedback. We make a lot of enhancements regularly, and here you will find the key features we've completed, versioned release notes, and what we're working on next.

⚠ Known Security Issue

A critical security vulnerability has been found in all versions of NXRM 3 up to and including 3.21.1. For details, please see [CVE-2020-10199²¹](#), [CVE-2020-10203²²](#), [CVE-2020-10204²³](#).

Sonatype recommends that administrators upgrade to 3.21.2 or later immediately.

Key Features

Yum Proxy Repositories

You can now define [Yum proxy repositories](#)(see page 460) with Nexus Repository Manager 3. We plan to enhance hosted and group repository support for this new format in the future. Though our intention is to build Yum support to be platform independent so that it will have its standalone dependencies on the external createrepo program.

⚠ Upgrading Yum proxy repositories from Nexus Repository Manager 2 to Nexus Repository Manager 3 is currently not supported.

Latest Release Notes

- [2020 Release Notes](#)(see page 39)
- [2019 Release Notes](#)(see page 44)
- [2018 Release Notes](#)(see page 68)
- [2017 Release Notes](#)(see page 85)
- [2016 Release Notes](#)(see page 110)
- [Milestone Releases](#)(see page 121)

Upgrading Repository Manager 2 to 3

If you're upgrading from Repository Manager 2.x, you must first upgrade your installation to [2.14.5](#)(see page 0). See the [upgrade compatibility matrix](#)(see page 146) for more information.

²¹ <https://support.sonatype.com/hc/en-us/articles/360044882533>

²² <https://support.sonatype.com/hc/en-us/articles/360044361594>

²³ <https://support.sonatype.com/hc/en-us/articles/360044882533>

2020 Release Notes

⚠ Security Fix

A critical security vulnerability has been found in all versions of NXRM 3 up to and including 3.21.1. For details, please see [CVE-2020-10199²⁴](#), [CVE-2020-10203²⁵](#), [CVE-2020-10204²⁶](#).

Sonatype recommends that administrators upgrade to 3.21.2 or newer immediately.

Repository Manager 3.22.0

2020-03-27

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.22.0. The issues fixed in this release can be found below. See the [complete release notes²⁷](#) for all resolved issues.

New and Noteworthy

SAML Authentication support

[NEXUS-20939²⁸](#)

Nexus Repository Manager allows users to authenticate with SAML identity providers.

Conan Format support

[NEXUS-14310²⁹](#)

Nexus Repository Manager now provides native support for proxying Conan repositories. Conan is a C/C++ package manager.

²⁴ <https://support.sonatype.com/hc/en-us/articles/360044882533>

²⁵ <https://support.sonatype.com/hc/en-us/articles/360044361594>

²⁶ <https://support.sonatype.com/hc/en-us/articles/360044882533>

²⁷ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=18933>

²⁸ <https://issues.sonatype.org/browse/NEXUS-20939>

²⁹ <https://issues.sonatype.org/browse/NEXUS-14310>

General Improvements

REST

- [NEXUS-21910³⁰] Additional REST provisioning support for npm, NuGet and PyPI repositories
- [NEXUS-22896³¹] performance regression in search REST API

Security

- [NEXUS-5716³²] All signed-in users can be assigned a default role
- [NEXUS-23272³³] Inability to add * permission to user on 3.21.2
- [NEXUS-16159³⁴] "Require user tokens for repository authentication" now enforced properly

npm

- [NEXUS-22245³⁵] Cannot delete npm scoped folder via UI

NuGet

- [NEXUS-23048³⁶] Allow proxying NuGet packages hosted by GitHub Packages

PyPI

- [NEXUS-22770³⁷] Change in stored PyPI proxy package paths creates duplicate assets and breaks browse node creation

Tasks

- [NEXUS-22054³⁸] "Repair - reconcile component database from blob store" task does not remove invalid component db references.
- [NEXUS-22729³⁹] Cleanup Policy task results in removal of maven-metadata from non-timestamped snapshots

³⁰ <https://issues.sonatype.org/browse/NEXUS-21910>

³¹ <https://issues.sonatype.org/browse/NEXUS-22896>

³² <https://issues.sonatype.org/browse/NEXUS-5716>

³³ <https://issues.sonatype.org/browse/NEXUS-23272>

³⁴ <https://issues.sonatype.org/browse/NEXUS-16159>

³⁵ <https://issues.sonatype.org/browse/NEXUS-22245>

³⁶ <https://issues.sonatype.org/browse/NEXUS-23048>

³⁷ <https://issues.sonatype.org/browse/NEXUS-22770>

³⁸ <https://issues.sonatype.org/browse/NEXUS-22054>

³⁹ <https://issues.sonatype.org/browse/NEXUS-22729>

Repository Manager 3.21.2

2020-03-23

Contains fixes for security vulnerabilities, it is recommended that administrators running earlier versions upgrade immediately.

Disabled Groovy Scripting By Default

In order to make NXRM more secure, we have disabled Groovy scripting engine by default. This affects Groovy scripts as used through the REST API and through scheduled tasks.

For more information (including how to re-enable Groovy scripting), see [NEXUS-23205](#)⁴⁰.

[NEXUS-23146](#)⁴¹

Fixes a remote code execution vulnerability.

[NEXUS-23147](#)⁴²

Fixes a remote code execution vulnerability for users with administrator permissions.

[NEXUS-23148](#)⁴³

Fixes a stored cross-site scripting vulnerability

Repository Manager 3.21.1

2020-02-18

Removes a broken menu entry incorrectly appearing for some users.

Repository Manager 3.21.0

2020-02-18

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.21.0. The issues fixed in this release can be found below. See the [complete release notes](#)⁴⁴ for all resolved issues.

⁴⁰ <https://issues.sonatype.org/browse/NEXUS-23205>

⁴¹ <https://issues.sonatype.org/browse/NEXUS-23146>

⁴² <https://issues.sonatype.org/browse/NEXUS-23147>

⁴³ <https://issues.sonatype.org/browse/NEXUS-23148>

⁴⁴ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=18732>

New and Noteworthy

p2 Format support

[NEXUS-11730](#)⁴⁵

Nexus Repository Manager now provides native support for proxying p2⁴⁶ repositories. p2 is a technology for provisioning and managing Eclipse- and Equinox-based applications.

 **Note**

This implementation does not include any ability to migrate p2 repositories from NXRM2 to NXRM3. Interest for that feature is being tracked in [NEXUS-22824](#)⁴⁷.

Helm Format support

[NEXUS-13325](#)⁴⁸

Helm is the first application package manager running atop Kubernetes(k8s). It allows describing the application structure through convenient helm-charts and managing it with simple commands.

NuGet V3 Proxy support

[NEXUS-10886](#)⁴⁹

NuGet V3 Proxy support gives Nexus Repository Manager users access to the up-to-date V3 API. This is the first part of a wider initiative to bring full V3 support, group and hosted will follow in future releases.

Serve Yum GPG key URLs

[NEXUS-16251](#)⁵⁰

Nexus Repository Manager now provides a common facility to allow RPM clients to get GPG keys to verify package signatures in remote repositories.

⁴⁵ <https://issues.sonatype.org/browse/NEXUS-11730>

⁴⁶ <https://help.sonatype.com/repomanager3/formats/p2-repositories>

⁴⁷ <https://issues.sonatype.org/browse/NEXUS-22824>

⁴⁸ <https://issues.sonatype.org/browse/NEXUS-13325>

⁴⁹ <https://issues.sonatype.org/browse/NEXUS-10886>

⁵⁰ <https://issues.sonatype.org/browse/NEXUS-16251>

npm ping support

[NEXUS-13434](#)⁵¹

Nexus Repository Manager is providing [npm cli ping](#)⁵² support.

General Improvements

Docker

- [\[NEXUS-18186\]](#)⁵³ Disabling redeploy for a private Docker repo breaks the "latest" tag

Audit

- [\[NEXUS-21730\]](#)⁵⁴ Audit log does not log all attributes for repository change events

Blobstore, Scheduled Tasks

- [\[NEXUS-21329\]](#)⁵⁵ "Remove a member from a blob store group" task processes missing files in the source blob store

Cleanup

- [\[NEXUS-18905\]](#)⁵⁶ Cleanup tasks fail with "No search context found for id" error

Crowd

- [\[NEXUS-13306\]](#)⁵⁷ Usernames containing non URL safe characters cannot authenticate using the Crowd realm

NuGet

- [\[NEXUS-16009\]](#)⁵⁸ Browse tree for NuGet proxy repositories shows packages that are not locally cached

⁵¹ <https://issues.sonatype.org/browse/NEXUS-13434>

⁵² <https://docs.npmjs.com/cli/ping.html>

⁵³ <https://issues.sonatype.org/browse/NEXUS-18186>

⁵⁴ <https://issues.sonatype.org/browse/NEXUS-21730>

⁵⁵ <https://issues.sonatype.org/browse/NEXUS-21329>

⁵⁶ <https://issues.sonatype.org/browse/NEXUS-18905>

⁵⁷ <https://issues.sonatype.org/browse/NEXUS-13306>

⁵⁸ <https://issues.sonatype.org/browse/NEXUS-16009>

PyPI

- [NEXUS-22051⁵⁹] PyPI group merge is not case sensitive

R

- [NEXUS-22351⁶⁰] R PACKAGES file lost on upgrade to 3.20.x

RubyGems

- [NEXUS-17477⁶¹] Unable to install hosted gem which has multiple version requirements

Yum

- [NEXUS-22052⁶²] Yum Metadata not rebuilt after staging deletion of rpm

2019 Release Notes

Repository Manager 3.20.1

2019-12-20

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.20.1. The issues fixed in this release can be found below.

Potential startup failure due to race condition loading product license in 3.20.0

NEXUS-22249⁶³

An error may occur when starting NXRM Pro due to race condition around the license loading.

OCommandExecutionException thrown during upgrade

NEXUS-22241⁶⁴

⁵⁹ <https://issues.sonatype.org/browse/NEXUS-22051>

⁶⁰ <https://issues.sonatype.org/browse/NEXUS-22351>

⁶¹ <https://issues.sonatype.org/browse/NEXUS-17477>

⁶² <https://issues.sonatype.org/browse/NEXUS-22052>

⁶³ <https://issues.sonatype.org/browse/NEXUS-22249>

⁶⁴ <https://issues.sonatype.org/browse/NEXUS-22241>

During the upgrade process from older releases (3.19.0 and before), NXRM may throw an exception when updating PyPI database configuration.

Repository Manager 3.20.0

- ⚠ A bug, [NEXUS-22241](#)⁶⁵, which affects upgrading your instance from 3.19.0 and earlier exists. You can either upgrade to 3.19.1 first or upgrade directly to 3.20.1+.

2019-12-13

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.20.0. A summary of the highlights in this release is shown below. See the [complete release notes](#)⁶⁶ for all resolved issues.

New and Noteworthy

jetty-http-redirect-to-https.xml removed and its use discouraged

[NEXUS-20269](#)⁶⁷

Following security best practices the example configuration file that redirects inbound HTTP requests to HTTPS requests has been removed from the distribution. Startup will fail if your configuration references the file and it cannot be found.

Should it still be needed, [instructions for adding it back are available](#)⁶⁸.

Ability to Clean Up by "Never downloaded"

[NEXUS-19424](#)⁶⁹

In past versions of cleanup, you could clean up by Last Downloaded but components with no download date would be ignored. This new function allows the ability to include in Cleanup items that have never been downloaded. In this case, uploaded date will be checked instead. In addition to the ticket, see [Cleanup Policies documentation](#)(see page 300) for more information.

⁶⁵ <https://issues.sonatype.org/browse/NEXUS-22241>

⁶⁶ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=18521>

⁶⁷ <https://issues.sonatype.org/browse/NEXUS-20269>

⁶⁸ <https://support.sonatype.com/hc/en-us/articles/360037845633>

⁶⁹ <https://issues.sonatype.org/browse/NEXUS-19424>

R Format Support

[NEXUS-9837](#)⁷⁰

Nexus Repository Manager now provides native support for R⁷¹ repositories. R is a language used for statistical analysis and machine learning.

Yum and CentOS 8

[NEXUS-21311](#)⁷²

Since the last NXRM release, CentOS 8 was released and it was found that NXRM did not function correctly with it. Full support is now added.

Enhanced npm login for private repository usages

[NEXUS-12456](#)⁷³

Added support for "npm login" bearer token authentication to proxied upstream npm private repositories.

npm whoami support

[NEXUS-13433](#)⁷⁴

Added support for "npm whoami".

npm package metadata does not return correct status code sometimes

[NEXUS-21371](#)⁷⁵

Conditional request handling added to npm group repositories to prevent errant 200 status codes from showing as 304s.

HSTS enabled by default for Inbound Jetty HTTPS connectors

[NEXUS-20268](#)⁷⁶

Following security best practices HSTS is now enabled by default if Jetty based HTTPS connectors (jetty-https.xml) are used.

⁷⁰ <https://issues.sonatype.org/browse/NEXUS-9837>

⁷¹ <https://help.sonatype.com/repomanager3/formats/r-repositories>

⁷² <https://issues.sonatype.org/browse/NEXUS-21311>

⁷³ <https://issues.sonatype.org/browse/NEXUS-12456>

⁷⁴ <https://issues.sonatype.org/browse/NEXUS-13433>

⁷⁵ <https://issues.sonatype.org/browse/NEXUS-21371>

⁷⁶ <https://issues.sonatype.org/browse/NEXUS-20268>

An [article has been published](#)⁷⁷ which describes how to enable it in older versions, where the configuration is to disable it, and how to modify the behavior.

General Improvements

Blobstore, UI

- [\[NEXUS-19811\]](#)⁷⁸ Offline and misconfigured blob store should be noted in UI

Cocoapods

- [\[NEXUS-21368\]](#)⁷⁹ Proxy repository removes get-params from HTTP sources

Content Selectors, UI

- [\[NEXUS-22144\]](#)⁸⁰ Slow performance displaying content selectors in UI

Docker

- [\[NEXUS-21306\]](#)⁸¹ Cannot proxy Docker repository on Bintray

Docker, Scheduled Tasks

- [\[NEXUS-21315\]](#)⁸² Extremely slow processing in "Docker - Delete unused manifests and images" task

Firewall

- [\[NEXUS-21672\]](#)⁸³ Group repo with proxy repo member to remote group repo responds 404 when remote group repo responds "403 Requested item is quarantined"

PyPI

- [\[NEXUS-18117\]](#)⁸⁴ PyPI ignoring python_requires metadata
- [\[NEXUS-20705\]](#)⁸⁵ Index can contain absolute URLs which bypass Nexus Repository Manager

⁷⁷ <https://support.sonatype.com/hc/en-us/articles/360024943193>

⁷⁸ <https://issues.sonatype.org/browse/NEXUS-19811>

⁷⁹ <https://issues.sonatype.org/browse/NEXUS-21368>

⁸⁰ <https://issues.sonatype.org/browse/NEXUS-22144>

⁸¹ <https://issues.sonatype.org/browse/NEXUS-21306>

⁸² <https://issues.sonatype.org/browse/NEXUS-21315>

⁸³ <https://issues.sonatype.org/browse/NEXUS-21672>

⁸⁴ <https://issues.sonatype.org/browse/NEXUS-18117>

⁸⁵ <https://issues.sonatype.org/browse/NEXUS-20705>

Repository Health Check

- [NEXUS-21589⁸⁶] Repository health check can fail if the same asset exists in more than one repository

REST

- [NEXUS-14233⁸⁷] Support managing Realms via the REST API

Scheduled Tasks, Maven

- [NEXUS-21138⁸⁸] Snapshot remover leaves maven-metadata.xml files deleted for a long time, breaking builds

SSL

- [NEXUS-12488⁸⁹] Remote https repository with TLS client certificate loaded in NXRM JVM keystore not trusted

Support Tools, UI

- [NEXUS-20140⁹⁰] 500 Server Error shown in Chrome console when accessing Support Status page

Repository Manager 3.19.1

2019-10-09

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.19.1. The issue fixed in this release can be found below.

Docker

NEXUS-21381⁹¹

Prevent Docker Proxy Repository throwing null pointer exceptions and blocking some image pulls after upgrade.

⁸⁶ <https://issues.sonatype.org/browse/NEXUS-21589>

⁸⁷ <https://issues.sonatype.org/browse/NEXUS-14233>

⁸⁸ <https://issues.sonatype.org/browse/NEXUS-21138>

⁸⁹ <https://issues.sonatype.org/browse/NEXUS-12488>

⁹⁰ <https://issues.sonatype.org/browse/NEXUS-20140>

⁹¹ <https://issues.sonatype.org/browse/NEXUS-21381>

Repository Manager 3.19.0

2019-09-30

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.19.0. A summary of the highlights in this release is shown below. See the [complete release notes⁹²](#) for all improvements and resolved issues.

New and Noteworthy

CocoaPods Format Support

[NEXUS-19970⁹³](#)

Nexus Repository Manager now provides native support for [CocoaPods\(see page 376\)](#) proxy repositories. Developers using Swift and Objective-C languages will now have easy access to pods and podspecs while gaining several advantages of proxy support.

Conda Format Support

[NEXUS-19866⁹⁴](#)

Nexus Repository Manager now provides native support for [Conda\(see page 381\)](#) proxy repositories. Those using Python, R, Ruby, Lua, Scala, Java, JavaScript, C/C++ and many more languages can now set up proxy repositories for Conda packages.

We are continuously getting feedback from developers and excited to continue the rapid growth of native support for our official [repository formats⁹⁵](#).

npm dist-tag

[NEXUS-9862⁹⁶](#)

New abilities for adding and removing "distribution tags" into npm metadata via the npm CLI. As one of our top requested improvements to Nexus Repository Manager, we are excited to continue our support for npm developers.

⁹² <https://issues.sonatype.org/secure/ReleaseNote.jspa?version=18222&projectId=10001>

⁹³ <https://issues.sonatype.org/browse/NEXUS-19970>

⁹⁴ <https://issues.sonatype.org/browse/NEXUS-19866>

⁹⁵ <https://www.sonatype.com/languages-packages>

⁹⁶ <https://issues.sonatype.org/browse/NEXUS-9862>

S3 Performance Improvements

[NEXUS-17797](#)⁹⁷

Higher performance S3 storage ability for Nexus Repository Manager deployments in AWS. This batch of work improves performance with enhanced support for encrypted S3 buckets, use of custom encryption keys, simplified permission testing, and essential improvements to storage space metrics.

Docker Windows Image Support

[NEXUS-19120](#)⁹⁸

Added support for Docker [Foreign Layers](#)(see page 400). Nexus Repository Manager users can now proxy docker images with foreign layers when pulling Microsoft Windows images.

REST API Improvements

[NEXUS-19144](#)⁹⁹, [NEXUS-19142](#)¹⁰⁰, [NEXUS-19143](#)¹⁰¹, [NEXUS-19145](#)¹⁰², [NEXUS-19146](#)¹⁰³, [NEXUS-16734](#)¹⁰⁴

Enhanced REST API endpoints for initial provisioning and maintenance of Nexus Repository Manager. Users will also benefit from new improvements to endpoints for local and external roles and permissions, privileges, and content selectors.

Go Format Data Integration

[NEXUS-20682](#)¹⁰⁵

Users who also have access to [Nexus Lifecycle](#)¹⁰⁶ can now view Go component details in the [Component IQ](#)¹⁰⁷ tab. Delivering advanced component information through [Nexus Intelligence](#)¹⁰⁸, details for Go components include information on policy violations, license issues, and security vulnerabilities that are known for a specific component.

⁹⁷ <https://issues.sonatype.org/browse/NEXUS-17797>

⁹⁸ <https://issues.sonatype.org/browse/NEXUS-19120>

⁹⁹ <https://issues.sonatype.org/browse/NEXUS-19144>

¹⁰⁰ <https://issues.sonatype.org/browse/NEXUS-19142>

¹⁰¹ <https://issues.sonatype.org/browse/NEXUS-19143>

¹⁰² <https://issues.sonatype.org/browse/NEXUS-19145>

¹⁰³ <https://issues.sonatype.org/browse/NEXUS-19146>

¹⁰⁴ <https://issues.sonatype.org/browse/NEXUS-16734>

¹⁰⁵ <https://issues.sonatype.org/browse/NEXUS-20682>

¹⁰⁶ <https://www.sonatype.com/product-nexus-lifecycle>

¹⁰⁷ <https://help.sonatype.com/integrations/iq-server-and-repository-management/iq-server-and-nxrm-3.x/viewing-component-information-in-nxrm-3.x>

¹⁰⁸ <https://www.sonatype.com/nexus-intelligence>

Multi-policy Cleanup

[NEXUS-19525](#)¹⁰⁹

NXRM admins can now make use of regular expressions to cleanup a given repository by team and/or project. The ability to apply multiple cleanup policies for an individual repository can even be used for longer retention of a certain set of artifacts.

General Improvements

NPM

- [\[NEXUS-10679\]](#)¹¹⁰ NPM repos don't handle HEAD requests

Docker

- [\[NEXUS-19102\]](#)¹¹¹ Unable to proxy private Azure (ACR) registry

Repository Manager 3.18.1

2019-08-09

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.18.1. The issues fixed in this release can be found below.

User Interface

[NEXUS-20674](#)¹¹²

Fixed an exception occurring when a user manually logs out of the user interface.

Firewall

[NEXUS-19235](#)¹¹³

Propagates the quarantine status code when NXRM proxies another NXRM with Firewall enabled.

¹⁰⁹ <https://issues.sonatype.org/browse/NEXUS-19525>

¹¹⁰ <https://issues.sonatype.org/browse/NEXUS-10679>

¹¹¹ <https://issues.sonatype.org/browse/NEXUS-19102>

¹¹² <https://issues.sonatype.org/browse/NEXUS-20674>

¹¹³ <https://issues.sonatype.org/browse/NEXUS-19235>

Repository Manager 3.18.0

2019-07-26

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.18.0. A summary of the highlights in this release is shown below. See the [complete release notes](#)¹¹⁴ for all resolved issues.

- i** Note: the permission model for browsing has been changed to behave similarly to NXRM2, see [NEXUS-20453](#)¹¹⁵ for details.

- ⚠** Our [Docker image](#)¹¹⁶ was updated to use Red Hat Universal Base Image (UBI). This can have an impact on outbound SSL connections, see here for details: [Sonatype Docker image security cryptographic standards may affect outbound TLS connections](#)¹¹⁷.

Security

[NEXUS-20479](#)¹¹⁸

Fixes two cross-site scripting (XSS) vulnerabilities.

Browse Performance

[NEXUS-20453](#)¹¹⁹

Improves browse performance particularly when the user has a large number of content selectors.

- i** The table that drives the repository browse tree will be rebuilt after upgrading to version 3.18.0. This will make the UI appear empty for a short period of time.

¹¹⁴ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=18142>

¹¹⁵ <https://issues.sonatype.org/browse/NEXUS-20453>

¹¹⁶ <https://github.com/sonatype/docker-nexus3>

¹¹⁷ <https://support.sonatype.com/hc/en-us/articles/360037535013-Sonatype-Docker-image-security-cryptographic-standards-may-affect-outbound-TLS-connections>

¹¹⁸ <https://issues.sonatype.org/browse/NEXUS-20479>

¹¹⁹ <https://issues.sonatype.org/browse/NEXUS-20453>

Database Backup & Upgrade Speed

[NEXUS-20100](#)¹²⁰, [NEXUS-20104](#)¹²¹

Optimized compression level and buffer size defaults of Orient database backups, resulting in faster upgrade speeds.

Default Memory Configuration

[NEXUS-19954](#)¹²²

Increased the default maximum heap and direct memory sizes for Nexus Repository Manager 3. Users will now receive increased sizing profiles and enhanced performance without having to manually change default configurations with new version upgrades.

Repository Manager 3.17.0

2019-06-24

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.17.0. A summary of the highlights in this release is shown below.

See the [complete release notes](#)¹²³ for all resolved issues.

Default admin password

You will find that the default password for the admin user is now randomly generated on startup of a new NXRM instance; see the [Post Install Checklist](#)¹²⁴ for more details.

¹²⁰ <https://issues.sonatype.org/browse/NEXUS-20100>

¹²¹ <https://issues.sonatype.org/browse/NEXUS-20104>

¹²² <https://issues.sonatype.org/browse/NEXUS-19954>

¹²³ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17890>

¹²⁴ <https://help.sonatype.com/repomanager3/installation/post-install-checklist>

⚠ HA-C Upgrade Changes

The contents of the `$install-dir/etc/fabric/hazelcast-network-default.xml` file have changed - you must update your own `$data-dir/etc/fabric/hazelcast-network.xml` to include the new section.

Also, if you're using HA-C TCP/IP discovery an additional configuration property is needed in `$data-dir/etc/nexus.properties`. See the [HA TCP/IP Discovery](#)¹²⁵ documentation for details.

New and Noteworthy

Routing Rules

For users with strict security controls we've added a feature we've called Routing Rules which can be used to prevent requests from being sent to external proxy repositories. This can for example allow users to block requests containing codenames from leaking to external repositories.

Apt Format Support

As part of our widespread format support we've begun to move one of the plugins from our great community members which supports the [apt format](#)(see page 367) into the core product. (Limited to single nodes in this release.)

Go Format Support

Supporting the burgeoning Go community we've added native support for [go repositories](#)(see page 406). (Limited to single nodes in this release.)

Beta Provisioning REST API

We've continued to add to our REST API with [new endpoints](#)(see page 496) to support operations on users, and user tokens. In this release they are flagged as beta, please take a look at our API documentation in the administration part of the user interface and submit feedback.

¹²⁵<https://help.sonatype.com/repomanager3/high-availability/configuring-hazelcast#ConfiguringHazelcast-TCP/IPDiscovery>

Setup Wizard

New instances as well as instances which have not changed anonymous configuration will prompt administrators upon logging in to confirm that anonymous users should be allowed to access the system. Similarly if the admin user is using the default password then upon logging in they will be required to change the password.

Removed Recent Connections Accesslog Database

The "accesslog" database which tracks recent unique connections has been removed, along with the *Recent Connections* section of the administration interface. This removal in no way impacts the request.log text file.

Reduced Logging for Missing Task Listeners

[NEXUS-19224](#)¹²⁶

Unnecessary logging of missing task listeners has been reduced.

Repository Manager 3.16.2

2019-05-31

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.16.2.

 The only fix in this release is for users of high availability.

Log entries for errors of "Task Already Running"

[NEXUS-19851](#)¹²⁷

Users of high availability may encounter logged errors that a task failed to start as it was already running.

Repository Manager 3.16.1

2019-04-16

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.16.1 . The resolved issues are shown below.

¹²⁶ <https://issues.sonatype.org/browse/NEXUS-19224>

¹²⁷ <https://issues.sonatype.org/browse/NEXUS-19851>

License installation

[NEXUS-19675](#)¹²⁸

After installing license (switching from OSS to PRO only, will not affect installing license in existing PRO instance) application doesn't start

Application Upgrade

[NEXUS-19703](#)¹²⁹

Upgrading to 3.16.0 PRO can fail when usertoken database contains entries with duplicate (case-insensitive) usernames

Repository Manager 3.16.0

2019-04-11

- ⚠ A bug which affects installing a license and moving from Nexus Repository OSS to Nexus Repository Pro has been discovered in version 3.16.0. This issue does not affect existing Nexus Repository Pro installations which are being upgraded to version 3.16.0 and has been fixed in 3.16.1.

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.16.0. A summary of the highlights in this release is shown below.

See the [complete release notes](#)¹³⁰ for all resolved issues.

New and Noteworthy

REST Node Status Endpoint

[NEXUS-18949](#)¹³¹

If you are running a cluster, or otherwise monitoring node health, you need a simple check to determine if your node should receive requests. To meet that need, we have added a new REST endpoint, /v1/status/writable. This endpoint returns a 200 status code when the node is ready to process requests. It's perfect

¹²⁸ <https://issues.sonatype.org/browse/NEXUS-19675>

¹²⁹ <https://issues.sonatype.org/browse/NEXUS-19703>

¹³⁰ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17837>

¹³¹ <https://issues.sonatype.org/browse/NEXUS-18949>

for integration with load balancers to remove a node from the active set when it becomes unhealthy and return it after the node has been recovered. See the documentation [Status API](#)(see page 509) for more.

Support Zip Improvements

[NEXUS-18711](#)¹³²

Getting all of the necessary support zips can be arduous in a HA-C environment. Administrators no longer have to access each node to generate support zips. Generating support zips will have the option to generate them on all nodes instead, and the pop up window will have a tab per node.

Previously, gathering support zips was a GUI driven process. So administrators could not use scripts to automate gathering support zips. The new REST endpoint, /v1/support/supportzip, gives programmatic access to support zips. The documentation for this API is here: [Support API](#)(see page 510)

HA-C topologies have whole new classes of metrics that are beneficial to gain insight into your cluster. A new log file, *nexus_cluster.log*, has been added to contain them. This log contains information about the cluster's health, such as internode latency and replication queue depth. Support zips will include this log to make introspection on the history of cluster health easier.

Additional System Status Checks for Professional HA Clusters

[NEXUS-18776](#)¹³³, [NEXUS-19313](#)¹³⁴, [NEXUS-18181](#)¹³⁵

System Status Checks are a great way for components of NXRM to report vital information to administrators. To that end, the checks have been added to the banner. In normal operation, every check is passing and administrators will see a green checkmark. However, if at least one status check is failing then the header will have a red exclamation point.

Three new status checks were added specifically targeting high availability health.

- *Cluster Size* is a new status check that ensures that your cluster has 3 nodes. If a cluster is missing a node, it can still respond to requests and save new components, but it is at risk of losing one more node and becoming inoperable.
- *Transactions* status check monitors an instance for any transaction with excessive retries in the last hour. Excess transactions can indicate things like heavy cluster load, inter-node network issue, or something more malignant.
- *Lifecycle Phase* is a status check that verifies that the node is running the proper lifecycle phase. Running in the wrong phase can occur due to startup issues, improper use of the lifecycle API, etc.

¹³² <https://issues.sonatype.org/browse/NEXUS-18711>

¹³³ <https://issues.sonatype.org/browse/NEXUS-18776>

¹³⁴ <https://issues.sonatype.org/browse/NEXUS-19313>

¹³⁵ <https://issues.sonatype.org/browse/NEXUS-18181>

Lifecycle API

[NEXUS-18183](#)¹³⁶

When a node becomes unhealthy, we now have a REST API to control the lifecycle phase of a NXRM node. A lifecycle phase is a step in the start up process used to group similar components and ensure that their dependencies are started before them. The REST API documentation has more information at [Lifecycle API\(see page 478\)](#).

Maintenance API

[NEXUS-18182](#)¹³⁷

The Maintenance API helps NXRM administrators understand their databases' status and recover from some error cases. Each endpoint can optionally take a node Id determining on which node the request is executed. See the REST API documentation for more insights at [Maintenance API\(see page 480\)](#).

REST Search Improvements

[NEXUS-14407](#)¹³⁸

REST search now supports sorting results, as well as filtering prerelease components, allowing the search and download endpoint to return the **latest** version of an asset. [More information in a community post](#)¹³⁹.

Update packaged JRE in installers to OpenJDK

[NEXUS-19619](#)¹⁴⁰

Mac and Windows installers now include the Zulu OpenJDK 8 JRE.

Improve stability of using HA-C with Nexus Firewall

[NEXUS-16852](#)¹⁴¹

Using Nexus Repository Manager in HA-C with Nexus Firewall could cause the cluster to have numerous concurrent modification exceptions. There were a number of changes made to both reduce the frequency of conflicts and more robustly handle them when they do occur.

¹³⁶ <https://issues.sonatype.org/browse/NEXUS-18183>

¹³⁷ <https://issues.sonatype.org/browse/NEXUS-18182>

¹³⁸ <https://issues.sonatype.org/browse/NEXUS-14407>

¹³⁹ <http://community.sonatype.com/t/nxrm-3-16-rest-search-and-filtering-enhancements/1586>

¹⁴⁰ <https://issues.sonatype.org/browse/NEXUS-19619>

¹⁴¹ <https://issues.sonatype.org/browse/NEXUS-16852>

Bower Resolver Updated

In addition to 3.16.0, we have also updated the bower resolver dependancies to the latest and most secure versions. We strongly recommend any users using bower update their bower resolver to the latest.

Metrics Timers Identifiers Have Changed

Some customers may be parsing the `/service/metrics/data` REST resource¹⁴² or utilizing the `info/metrics.json` file in support zips. This release updates our metrics library¹⁴³, which has changed the names of `timers` to include the suffix `'.timer'`. For example:

Old timer: `com.sonatype.nexus.plugins.healthcheck.ui.HealthCheckStatusComponent.read`

New timer: `com.sonatype.nexus.plugins.healthcheck.ui.HealthCheckStatusComponent.read.timer`

The "version" field of the metrics file reflects this change, revving the version from 3.0.0 to 4.0.0.

Audit Database Replaced With Log File

[NEXUS-18813](#)¹⁴⁴

The audit database at `$data-dir/db/audit` is no longer in use and is replaced with [audit log files](#)(see page 286) written and archived in `$data-dir/log/audit` . Administrators may remove the audit database files from disk at their own discretion.

General Improvements

Cleanup, Search

- [NEXUS-18905]¹⁴⁵ Cleanup tasks fail with "No search context found for id" error

Content Selectors

- [NEXUS-18509]¹⁴⁶ Make JEXL and CSEL behave the same for expressions without a leading slash

¹⁴² <https://support.sonatype.com/hc/en-us/articles/226254487-Nexus-3-Server-Metrics-and-Health-Information>

¹⁴³ <https://issues.sonatype.org/browse/NEXUS-18992>

¹⁴⁴ <https://issues.sonatype.org/browse/NEXUS-18813>

¹⁴⁵ <https://issues.sonatype.org/browse/NEXUS-18905>

¹⁴⁶ <https://issues.sonatype.org/browse/NEXUS-18509>

Content Selectors, Scripting, Upgrade

- [NEXUS-17850¹⁴⁷] API does not validate contents of content selectors. Invalid content selectors can lead to failed upgrade.

Database, Scheduled Tasks

- [NEXUS-18983¹⁴⁸] If NXRM is read-only or lacks quorum, then run now triggers make startup fail.

REST

- [NEXUS-18816¹⁴⁹] Slow delete performance when using REST API

Docker

- [NEXUS-19125¹⁵⁰] Docker pull from mcr.microsoft.com results in 403

HA, Repository

- [NEXUS-19229¹⁵¹] LastDownloadedHandler conflicts in HA can cause long-running retries/rollbacks

LDAP, User Token, Upgrade

- [NEXUS-13639¹⁵²] User tokens not migrated if LDAP user ID case does not match login case

LDAP, Tree View, Logging

- [NEXUS-17616¹⁵³] On browse w/ LDAP, if no perms, a bunch of warns are fired

Cleanup, Logging

- [NEXUS-18731¹⁵⁴] specifics about what is deleted by cleanup policies is not logged

¹⁴⁷ <https://issues.sonatype.org/browse/NEXUS-17850>

¹⁴⁸ <https://issues.sonatype.org/browse/NEXUS-18983>

¹⁴⁹ <https://issues.sonatype.org/browse/NEXUS-18816>

¹⁵⁰ <https://issues.sonatype.org/browse/NEXUS-19125>

¹⁵¹ <https://issues.sonatype.org/browse/NEXUS-19229>

¹⁵² <https://issues.sonatype.org/browse/NEXUS-13639>

¹⁵³ <https://issues.sonatype.org/browse/NEXUS-17616>

¹⁵⁴ <https://issues.sonatype.org/browse/NEXUS-18731>

Proxy Repository, Logging

- [NEXUS-17502]¹⁵⁵ Content validation message does not log which repository the invalid content is coming from

Maven

- [NEXUS-16853]¹⁵⁶ Enhance content validation for maven-metadata.xml files

Maven, Scheduled Tasks

- [NEXUS-19034]¹⁵⁷ NPE on "Purge unused snapshots" task

Tags

- [NEXUS-17908]¹⁵⁸ Tag association may intermittently fail for new artifact

Security, Upgrade

- [NEXUS-12222]¹⁵⁹ NXRM2 repository view privileges are not migrated to NXRM3 browse privileges during upgrade

NPM

- [NEXUS-18774]¹⁶⁰ allow scoped NPM package name parts that start with '.' or '_'
- [NEXUS-17896]¹⁶¹ concurrent requests for large npm metadata can lead to OutOfMemory during serialization performance

PyPI

- [NEXUS-19121]¹⁶² Delete of component or asset from PyPi proxy repository fails

¹⁵⁵ <https://issues.sonatype.org/browse/NEXUS-17502>

¹⁵⁶ <https://issues.sonatype.org/browse/NEXUS-16853>

¹⁵⁷ <https://issues.sonatype.org/browse/NEXUS-19034>

¹⁵⁸ <https://issues.sonatype.org/browse/NEXUS-17908>

¹⁵⁹ <https://issues.sonatype.org/browse/NEXUS-12222>

¹⁶⁰ <https://issues.sonatype.org/browse/NEXUS-18774>

¹⁶¹ <https://issues.sonatype.org/browse/NEXUS-17896>

¹⁶² <https://issues.sonatype.org/browse/NEXUS-19121>

Repository, UI

- [NEXUS-19118]^{163]} Clicking on the links in a repository for component/asset browse gives a 404

Repository Health Check

- [NEXUS-18950]^{164]} "Download trend" disabled text misleading

REST

- [NEXUS-14407]^{165]} REST Search & Download by 'Latest'

Staging

- [NEXUS-19085]^{166]} staging promotion move of more than 500 components may fail with IllegalStateException Unable to find component by id performance

UI

- [NEXUS-13057]^{167]} "Uncaught TypeError" UI errors
- [NEXUS-19051]^{168]} Edge and Internet Explorer errors in the user interface
- [NEXUS-12253]^{169]} Order by version incorrect

Upload UI

- [NEXUS-18277]^{170]} UI upload creates temporary files in java.io.tmpdir

Yum

- [NEXUS-16057]^{171]} Add UI upload for Yum
- [NEXUS-17884]^{172]} upload of source rpm fails in yum hosted
- [NEXUS-17920]^{173]} Deleting an rpm via DELETE to /repository does not update metadata

¹⁶³ <https://issues.sonatype.org/browse/NEXUS-19118>

¹⁶⁴ <https://issues.sonatype.org/browse/NEXUS-18950>

¹⁶⁵ <https://issues.sonatype.org/browse/NEXUS-14407>

¹⁶⁶ <https://issues.sonatype.org/browse/NEXUS-19085>

¹⁶⁷ <https://issues.sonatype.org/browse/NEXUS-13057>

¹⁶⁸ <https://issues.sonatype.org/browse/NEXUS-19051>

¹⁶⁹ <https://issues.sonatype.org/browse/NEXUS-12253>

¹⁷⁰ <https://issues.sonatype.org/browse/NEXUS-18277>

¹⁷¹ <https://issues.sonatype.org/browse/NEXUS-16057>

¹⁷² <https://issues.sonatype.org/browse/NEXUS-17884>

¹⁷³ <https://issues.sonatype.org/browse/NEXUS-17920>

Repository Manager 3.15.2

2019-01-29

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.15.2. A summary of the highlights in this release is shown below.

Search Hotfix

[NEXUS-18909](#)¹⁷⁴

Fix search filtering using repository query parameter, when group repository is entered (REST and UI).

Docker image

[NEXUS-19019](#)¹⁷⁵

Our official Docker image switches to OpenJDK as our recommended Java Runtime Environment. It's worth checking whether the difference has impact on your deployment (e.g., contains a different system trust store that may affect your SSL connections).

Repository Manager 3.15.1

2019-01-18

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.15.1. A summary of the highlights in this release is shown below.

NPM Hotfix

[NEXUS-18850](#)¹⁷⁶

Fix anti-cross-site request forgery token mismatch blocking valid npm client publish and login.

Repository Manager 3.15.0

2019-1-11

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.15.0. A summary of the highlights in this release is shown below.

¹⁷⁴ <https://issues.sonatype.org/browse/NEXUS-18909>

¹⁷⁵ <https://issues.sonatype.org/browse/NEXUS-19019>

¹⁷⁶ <https://issues.sonatype.org/browse/NEXUS-18850>

See the [complete release notes](#)¹⁷⁷ for all resolved issues.

i We have found an issue in this release that affects NPM users, as well as using the REST search filtering by repository group names. We recommend NPM and/or REST search users upgrade immediately to 3.15.2-01 (see [Download Archives](#)([see page 23](#)) for access) or newer.

New and Noteworthy

Beta REST API Endpoints Removed

The Nexus Repo REST API moved from beta to release status in the 3.13.10 release. The old /service/rest/beta endpoints were kept for compatibility in the 3.13.0 and 3.14.0 releases, but they have now been removed ([NEXUS-18562](#)¹⁷⁸). Update any scripts that use /service/rest/beta endpoints to use the new /service/rest/v1 endpoints.

Dynamic Storage

For our Professional licensed customers, we added a new level of flexibility for administrators when planning and updating their blob stores. Multiple blob stores can be combined into a Blob Store Group. These groups allow an individual repository to use storage across multiple locations and devices. Member blob stores can be added or removed from a group to ease the burden of migrations. Also added Fill Policies which give administrators more control and insight into how components are being stored.

For all our users (OSS and PRO) we have added Soft Quota which allow you to receive a warning when your blobstore has reached a configured metric.

For more detail, see our documentation [Storage Guide](#)([see page 281](#)).

OpenJDK 8 Runtime

[NEXUS-6300](#)¹⁷⁹

Users have been reporting OpenJDK has worked fine as the JVM for repository manager 3. As of 3.15.0, we now run our full test suite using a compliant Java SE standard distribution of OpenJDK 8. In light of the Oracle licensing changes to their Oracle JDK distributions, you can now be sure repository manager is fully tested against the standard.

UI Upload left files in temporary directory after upload was complete

[NEXUS-18276](#)¹⁸⁰

¹⁷⁷ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17768>

¹⁷⁸ <https://issues.sonatype.org/browse/NEXUS-18562>

¹⁷⁹ <https://issues.sonatype.org/browse/NEXUS-6300>

¹⁸⁰ <https://issues.sonatype.org/browse/NEXUS-18276>

This bug would cause temp files to be left behind when the UI Upload feature was used. This could cause eventual disk space issues that would not be cleaned up by the standard NXRM cleanup tasks. This ticket fixes the issue going forward but does not assume to delete anything from the file system. If you utilize Upload UI frequently, we recommend checking the temp directory to see if you can get some disk space back.

Use HTTPS for outreach Base URL

[NEXUS-18271](#)¹⁸¹

In order to promote security, we changed our *Welcome Outreach* capability to use HTTPS instead of HTTP.

Analytics plugin deprecated

Due to complications, the Analytics plugin and respective pages within the application have been removed from the system. If it is restored, we will have a new feature release note.

General Improvements

Blobstore

- [\[NEXUS-18252\]](#)¹⁸² Repository manager will not start on blobstore problems or errors

Maven

- [\[NEXUS-15095\]](#)¹⁸³ MissingBlobException can occur when publishing Maven index
- [\[NEXUS-18196\]](#)¹⁸⁴ ArrayIndexOutOfBoundsException when uploading large POM

Docker

- [\[NEXUS-12684\]](#)¹⁸⁵ HEAD request to /v2/<name>/manifests/<reference> results in 404 error
- [\[NEXUS-18263\]](#)¹⁸⁶ Docker proxy repositories configured with a remote URL including extra path info will not proxy correctly
- [\[NEXUS-18353\]](#)¹⁸⁷ Can't proxy older gcr.io Docker images

¹⁸¹ <https://issues.sonatype.org/browse/NEXUS-18271>

¹⁸² <https://issues.sonatype.org/browse/NEXUS-18252>

¹⁸³ <https://issues.sonatype.org/browse/NEXUS-15095>

¹⁸⁴ <https://issues.sonatype.org/browse/NEXUS-18196>

¹⁸⁵ <https://issues.sonatype.org/browse/NEXUS-12684>

¹⁸⁶ <https://issues.sonatype.org/browse/NEXUS-18263>

¹⁸⁷ <https://issues.sonatype.org/browse/NEXUS-18353>

UI

- [NEXUS-18100¹⁸⁸] LDAP UI does not reflect configuration
- [NEXUS-13626¹⁸⁹] Made Privileges box wider so Privileges fit
- [NEXUS-7996¹⁹⁰] Changing dropdown does not show users list until refresh

LDAP

- [NEXUS-18564¹⁹¹] Delete orphaned API keys task run before any other HTTP activity can stop some LDAP operations

RubyGems, Upgrade

- [NEXUS-16964¹⁹²] Upgrade from 2.14.8 to 3.10.0 may prevent download of rubygems hosted repository gems

NuGet

- [NEXUS-17501¹⁹³] Caching of NuGet metadata causes thread serialization, query slowdowns under load performance

PyPI

- [NEXUS-14465¹⁹⁴] PyPI hosted repository does not send etag header
- [NEXUS-17903¹⁹⁵] PyCharm does not work with PyPI repository
- [NEXUS-18187¹⁹⁶] PyPI proxy of https://bloomberg.bintray.com/pip does not work
- [NEXUS-16401¹⁹⁷] PyPI hosted repository packages can only be searched by pep-0503 normalized name

¹⁸⁸ <https://issues.sonatype.org/browse/NEXUS-18100>

¹⁸⁹ <https://issues.sonatype.org/browse/NEXUS-13626>

¹⁹⁰ <https://issues.sonatype.org/browse/NEXUS-7996>

¹⁹¹ <https://issues.sonatype.org/browse/NEXUS-18564>

¹⁹² <https://issues.sonatype.org/browse/NEXUS-16964>

¹⁹³ <https://issues.sonatype.org/browse/NEXUS-17501>

¹⁹⁴ <https://issues.sonatype.org/browse/NEXUS-14465>

¹⁹⁵ <https://issues.sonatype.org/browse/NEXUS-17903>

¹⁹⁶ <https://issues.sonatype.org/browse/NEXUS-18187>

¹⁹⁷ <https://issues.sonatype.org/browse/NEXUS-16401>

REST

- [NEXUS-18345¹⁹⁸] Running a "/search/assets" REST API call with just a "repository" query parameter does not give full results

Scheduled Tasks

- [NEXUS-12828¹⁹⁹] Submitting more than 20 tasks at once causes ERROR for some tasks

Content Selectors, Tree View

- [NEXUS-15085²⁰⁰] Tree View is slow when there are large numbers of content selector privileges

Tree View

- [NEXUS-16384²⁰¹] Maven SNAPSHOT timestamp versioned files are not direct children to the base snapshot version in tree or html view
- [NEXUS-14682²⁰²] Support deleting all assets under entire selected tree nodes
- [NEXUS-15179²⁰³] Deleting last pom leaves folder shell

UI

- [NEXUS-18119²⁰⁴] New role not available for use until role page revisited
- [NEXUS-12100²⁰⁵] Delete button on component and asset is active even without permission to do so

Upload UI

- [NEXUS-18276²⁰⁶] UI upload leaves files in temporary directory after upload is complete
- [NEXUS-18494²⁰⁷] UI upload fails if it takes more than 60 seconds

198 <https://issues.sonatype.org/browse/NEXUS-18345>

199 <https://issues.sonatype.org/browse/NEXUS-12828>

200 <https://issues.sonatype.org/browse/NEXUS-15085>

201 <https://issues.sonatype.org/browse/NEXUS-16384>

202 <https://issues.sonatype.org/browse/NEXUS-14682>

203 <https://issues.sonatype.org/browse/NEXUS-15179>

204 <https://issues.sonatype.org/browse/NEXUS-18119>

205 <https://issues.sonatype.org/browse/NEXUS-12100>

206 <https://issues.sonatype.org/browse/NEXUS-18276>

207 <https://issues.sonatype.org/browse/NEXUS-18494>

Webhooks

- [NEXUS-18299]²⁰⁸ JsonSyntaxException attempting to create Webhook capability

Yum

- [NEXUS-18261]²⁰⁹ Hosted yum metadata not rebuilding due to parsing issues in the path being queried

Search, Tree View

- [NEXUS-18617]²¹⁰ Disable asset download count feature in all new/upgraded installations

Maven, Search

- [NEXUS-17803]²¹¹ Allow search by g:a:v string

S3, Blobstore

- [NEXUS-18631]²¹² Allow multipart copy for AWS S3 blob storage

2018 Release Notes

Repository Manager 3.14.0

2018-10-12

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.14.0. A summary of the highlights in this release is shown below.

See the [complete release notes](#)²¹³ for all resolved issues.

²⁰⁸ <https://issues.sonatype.org/browse/NEXUS-18299>

²⁰⁹ <https://issues.sonatype.org/browse/NEXUS-18261>

²¹⁰ <https://issues.sonatype.org/browse/NEXUS-18617>

²¹¹ <https://issues.sonatype.org/browse/NEXUS-17803>

²¹² <https://issues.sonatype.org/browse/NEXUS-18631>

²¹³ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17626>

New and Noteworthy

Security Fixes for multiple CVEs

Multiple security vulnerabilities have been corrected in 3.14. For details, please see:

- [CVE-2018-16619²¹⁴](#)
- [CVE-2018-16620²¹⁵](#)
- [CVE-2018-16621²¹⁶](#)

Sonatype recommends that administrators running 3.13 and earlier upgrade immediately.

Cleanup Policies

This release introduces Cleanup Policies which are intended to eventually be a complete solution to cleanup old or unwanted components and replace a number of tasks which already exist. Briefly, you would create a cleanup policy for a format with the details of the criteria to cleanup. Then you'd assign the policy to a repository or repositories and then repeat creating and assigning as many times as needed. Cleanup itself runs as a task on a schedule determined by you doing the work automatically. This release focused on having a basic set of actions for all formats. See [here\(see page 297\)](#) for more detail.

Tags and Custom Attributes

[NEXUS-17694²¹⁷](#)

Tags and Custom Attributes are now in the UI. More details [here\(see page 232\)](#).

Search Improvements

[NEXUS-8884²¹⁸](#) [NEXUS-8798²¹⁹](#), [NEXUS-13227²²⁰](#), [NEXUS-12691²²¹](#)

A number of improvements have been made to the search functionality.

²¹⁴<https://support.sonatype.com/hc/en-us/articles/360010789893-CVE-2018-16619-Nexus-Repository-Manager-XSS-October-17-2018>

²¹⁵ <https://support.sonatype.com/hc/en-us/articles/360010789453-CVE-2018-16620-Nexus-Repository-Manager-Missing-Access-Controls-October-17-2018>

²¹⁶<https://support.sonatype.com/hc/en-us/articles/360010789153-CVE-2018-16621-Nexus-Repository-Manager-Java-Injection-October-17-2018>

²¹⁷ <https://issues.sonatype.org/browse/NEXUS-17694>

²¹⁸ <https://issues.sonatype.org/browse/NEXUS-8884>

²¹⁹ <https://issues.sonatype.org/browse/NEXUS-8798>

²²⁰ <https://issues.sonatype.org/browse/NEXUS-13227>

²²¹ <https://issues.sonatype.org/browse/NEXUS-12691>

Staging: more formats can move and delete

[NEXUS-16673](https://issues.sonatype.org/browse/NEXUS-16673)²²², [NEXUS-16674](https://issues.sonatype.org/browse/NEXUS-16674)²²³

Professional users can now move and delete components in NuGet and YUM hosted repositories

Maven metadata rebuild task performance improvement

[NEXUS-17759](https://issues.sonatype.org/browse/NEXUS-17759)²²⁴

The maven metadata rebuild task previously could be slow. The performance has been improved in this release.

Content selector permissions to staging operations

[NEXUS-17199](https://issues.sonatype.org/browse/NEXUS-17199)²²⁵

Content selectors can be used to allow staging operations on portions of repositories as opposed to the entire repo or nothing.

General Improvements

Browse Storage

- [\[NEXUS-17690\]](https://issues.sonatype.org/browse/NEXUS-17690)²²⁶ - Browse Docker Repository causes IllegalStateException 'Comparison method violates its general contract'

Repository Health Check

- [\[NEXUS-17741\]](https://issues.sonatype.org/browse/NEXUS-17741)²²⁷ - Resolve Repository Health Check invalid states post-restart

Upgrade

- [\[NEXUS-17772\]](https://issues.sonatype.org/browse/NEXUS-17772)²²⁸ - Migration from 2 to 3 leaves low heap memory ERROR in Nexus 3 performance
- [\[NEXUS-17460\]](https://issues.sonatype.org/browse/NEXUS-17460)²²⁹ - Nexus 2 to 3 upgrade fails with concurrency error in NuGet

²²² <https://issues.sonatype.org/browse/NEXUS-16673>

²²³ <https://issues.sonatype.org/browse/NEXUS-16674>

²²⁴ <https://issues.sonatype.org/browse/NEXUS-17759>

²²⁵ <https://issues.sonatype.org/browse/NEXUS-17199>

²²⁶ <https://issues.sonatype.org/browse/NEXUS-17690>

²²⁷ <https://issues.sonatype.org/browse/NEXUS-17741>

²²⁸ <https://issues.sonatype.org/browse/NEXUS-17772>

²²⁹ <https://issues.sonatype.org/browse/NEXUS-17460>

Docker

- [\[NEXUS-17548\]²³⁰](https://issues.sonatype.org/browse/NEXUS-17548) - Docker proxy repositories auto-block for images that don't exist

UI

- [\[NEXUS-12033\]²³¹](https://issues.sonatype.org/browse/NEXUS-12033) - Add method to configure request timeout via UI
- [\[NEXUS-14593\]²³²](https://issues.sonatype.org/browse/NEXUS-14593) - Errors reported when accessing UI via index.html
- [\[NEXUS-17259\]²³³](https://issues.sonatype.org/browse/NEXUS-17259) - NullPointerException and upload hangs using UI to upload a larger file to raw repository

HTML View

- [\[NEXUS-17614\]²³⁴](https://issues.sonatype.org/browse/NEXUS-17614) - HTML Browse View Encodes File Paths

NPM

- [\[NEXUS-16312\]²³⁵](https://issues.sonatype.org/browse/NEXUS-16312) - Metadata for NPM group considers pre-release version higher than actual version

REST

- [\[NEXUS-17921\]²³⁶](https://issues.sonatype.org/browse/NEXUS-17921) - REST API upload that fails due to lack of permissions returns 404 (not found)

NuGet

- [\[NEXUS-17611\]²³⁷](https://issues.sonatype.org/browse/NEXUS-17611) - 404s returned for packages containing build-metadata in version
- [\[NEXUS-17712\]²³⁸](https://issues.sonatype.org/browse/NEXUS-17712) - Nuget repo unable to proxy packages with 4 digit version

User Token

- [\[NEXUS-17498\]²³⁹](https://issues.sonatype.org/browse/NEXUS-17498) - Option to generate URL-safe user tokens for URL based authentication

²³⁰ <https://issues.sonatype.org/browse/NEXUS-17548>

²³¹ <https://issues.sonatype.org/browse/NEXUS-12033>

²³² <https://issues.sonatype.org/browse/NEXUS-14593>

²³³ <https://issues.sonatype.org/browse/NEXUS-17259>

²³⁴ <https://issues.sonatype.org/browse/NEXUS-17614>

²³⁵ <https://issues.sonatype.org/browse/NEXUS-16312>

²³⁶ <https://issues.sonatype.org/browse/NEXUS-17921>

²³⁷ <https://issues.sonatype.org/browse/NEXUS-17611>

²³⁸ <https://issues.sonatype.org/browse/NEXUS-17712>

²³⁹ <https://issues.sonatype.org/browse/NEXUS-17498>

Security

- [NEXUS-10692²⁴⁰] - Do not prompt for user credentials for RUT authenticated users

Yum

- [NEXUS-17886²⁴¹] - Yum metadata from unrelated folder incorrectly removed when regenerating a folder

Backup

- [NEXUS-17233²⁴²] - Restarting while backup is in progress leaves NXRM as read-only

Repository Manager 3.13.0

2018-07-19

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.13.0. A summary of the highlights in this release is shown below.

Complete [release notes](#)²⁴³ for all resolved issues.

New and Noteworthy

Request Log Line Format Change

[NEXUS-16903](#)²⁴⁴

The default request.log line format has changed to include the request Content-Length header value. External log parsers like Splunk may need adjustment to account for this change.

REST API v1

[NEXUS-17633](#)²⁴⁵

The REST API has been released under new v1 endpoints (/service/rest/v1/...). While the previous endpoints will remain available (e.g., /service/rest/beta/...) they are subject to change; it is highly recommended to

²⁴⁰ <https://issues.sonatype.org/browse/NEXUS-10692>

²⁴¹ <https://issues.sonatype.org/browse/NEXUS-17886>

²⁴² <https://issues.sonatype.org/browse/NEXUS-17233>

²⁴³ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17504>

²⁴⁴ <https://issues.sonatype.org/browse/NEXUS-16903>

²⁴⁵ <https://issues.sonatype.org/browse/NEXUS-17633>

update any integrations to leverage the new endpoint paths. Please see [REST and Integration API](#)(see page 464) for more information about the APIs.

General Improvements

Scheduled Tasks

- [\[NEXUS-9605\]](#)²⁴⁶ - Task last run and last result not persisted correctly
- [\[NEXUS-13121\]](#)²⁴⁷ - Tasks may appear as 'Starting' or 'Cancelling' indefinitely and cannot be stopped, cancelled, or deleted
- [\[NEXUS-17008\]](#)²⁴⁸ - Task will never run again if its previous run time passes its next scheduled start time
- [\[NEXUS-17262\]](#)²⁴⁹ - Removing repository does not remove tasks specific to the removed repository

Yum

- [\[NEXUS-16545\]](#)²⁵⁰ - Yum Metadata Generation performance improvements

HA

- [\[NEXUS-17440\]](#)²⁵¹ - "Unable to detect which node you are currently connected to" warning can appear in non-clustered instance

Maven Repository

- [\[NEXUS-16430\]](#)²⁵² - Connection reset when uploading large file using Apache Ivy

Upgrade

- [\[NEXUS-17455\]](#)²⁵³ - Last-Modified not returned in header for migrated RAW artifacts
- [\[NEXUS-16985\]](#)²⁵⁴ - Nexus 2 to 3 migration fails if there are staging build promotion repositories

²⁴⁶ <https://issues.sonatype.org/browse/NEXUS-9605>

²⁴⁷ <https://issues.sonatype.org/browse/NEXUS-13121>

²⁴⁸ <https://issues.sonatype.org/browse/NEXUS-17008>

²⁴⁹ <https://issues.sonatype.org/browse/NEXUS-17262>

²⁵⁰ <https://issues.sonatype.org/browse/NEXUS-16545>

²⁵¹ <https://issues.sonatype.org/browse/NEXUS-17440>

²⁵² <https://issues.sonatype.org/browse/NEXUS-16430>

²⁵³ <https://issues.sonatype.org/browse/NEXUS-17455>

²⁵⁴ <https://issues.sonatype.org/browse/NEXUS-16985>

NPM

- [NEXUS-15714²⁵⁵] - Continue to serve locally cached proxied npm packages that are unpublished on the remote

NuGet

- [NEXUS-16476²⁵⁶] - Do not change NuGet API key when a user's password is changed

Rubygems

- [NEXUS-16461²⁵⁷] - New rubygems dependency files are cached in blob storage every time Nexus requests them from a proxy repository remote

Security

- [NEXUS-17231²⁵⁸] - User role mappings will match user IDs case insensitively for LDAP, Crowd, and default authentication realms

Yum

- [NEXUS-16409²⁵⁹] - Support HTTP DELETE requests on RPMs to a Yum hosted repository

Repository Manager 3.12.1

6/11/2018

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.12.1. This is a patch release fixing a single urgent bug noted below.

Upload UI

- [NEXUS-17287²⁶⁰] - Maven UI/REST API upload results in empty pom

²⁵⁵ <https://issues.sonatype.org/browse/NEXUS-15714>

²⁵⁶ <https://issues.sonatype.org/browse/NEXUS-16476>

²⁵⁷ <https://issues.sonatype.org/browse/NEXUS-16461>

²⁵⁸ <https://issues.sonatype.org/browse/NEXUS-17231>

²⁵⁹ <https://issues.sonatype.org/browse/NEXUS-16409>

²⁶⁰ <https://issues.sonatype.org/browse/NEXUS-17287>

Repository Manager 3.12.0

5/22/2018

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.12.0. A summary of the highlights in this release is shown below.

See the [complete release notes](#)²⁶¹ for all resolved issues.

Upgrade to 3.12.1 Recommended

Version 3.12.0 has a critical bug regarding the Maven Upload UI/REST endpoint. When uploading a POM file (and not having it autogenerated), empty content is stored for the file. If you are using Maven repositories and plan on uploading files through the UI or REST interface, it is HIGHLY recommended to not use this version, rather use version 3.12.1 (or newer) where the issue is resolved.

New and Noteworthy

Built-in S3 Blobstore support

[NEXUS-11409](#)²⁶²

We've taken the popular [S3 Blobstore Plugin](#)²⁶³ and are now including it with OSS and PRO distributions.

General Improvements

Security

- [\[NEXUS-16980\]](#)²⁶⁴ - User tokens cannot be retrieved by users who have "nx-usertoken-current" privilege

²⁶¹ <https://issues.sonatype.org/secure/ReleaseNote.jspa?version=17493&projectId=10001>

²⁶² <https://issues.sonatype.org/browse/NEXUS-11409>

²⁶³ <https://github.com/sonatype/nexus-blobstore-s3>

²⁶⁴ <https://issues.sonatype.org/browse/NEXUS-16980>

Upload UI

- [NEXUS-16740²⁶⁵] - Upload interface doesn't update or create metadata after upload file

REST

- [NEXUS-16225²⁶⁶] - Swagger UI caching causing load problems on upgrade

NPM

- [NEXUS-11139²⁶⁷] - ConcurrentModificationException when deleting NPM resource

Docker

- [NEXUS-15582²⁶⁸] - docker proxy repository does not work for container-registry.oracle.com
- [NEXUS-16718²⁶⁹] - "scope" authentication errors when connecting to registry.connect.redhat.com
- [NEXUS-16992²⁷⁰] - 403 forbidden when a proxy repository authenticates to private docker registry in gitlab

Repository Manager 3.11.0

5/1/2018

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.11.0. A summary of the highlights in this release is shown below.

See the [complete release notes](#)²⁷¹ for all resolved issues.

New and Noteworthy

Restore Directory Location Changed

[NEXUS-14493](#)²⁷²

²⁶⁵ <https://issues.sonatype.org/browse/NEXUS-16740>

²⁶⁶ <https://issues.sonatype.org/browse/NEXUS-16225>

²⁶⁷ <https://issues.sonatype.org/browse/NEXUS-11139>

²⁶⁸ <https://issues.sonatype.org/browse/NEXUS-15582>

²⁶⁹ <https://issues.sonatype.org/browse/NEXUS-16718>

²⁷⁰ <https://issues.sonatype.org/browse/NEXUS-16992>

²⁷¹ <https://issues.sonatype.org/secure/ReleaseNote.jspa?version=17480&projectId=10001>

²⁷² <http://Yum Group>

We found old database restore location (`$data-dir/backup`) was causing confusion. The location has been changed, database backups should be placed in (`$data-dir/restore-from-backup`) for restoration as of the 3.11.0 release.

Yum Group

[NEXUS-12331](#)²⁷³

This release includes the ability to create Yum Group repositories, see the documentation [here](#)(see page 460) for further information.

Staging via REST API

[NEXUS-11446](#)²⁷⁴

Nexus Repository Manager PRO customers are now able to utilise REST API endpoints for staging requirements into their CI/CD pipeline. The REST API exposes tag, move and delete endpoints to accomplish this. See the documentation [here](#)(see page 352) for further information.

Upload UI

PRO customers now have the ability to tag components while uploading them through the UI.

General Improvements

Security

- [\[NEXUS-16227\]](#)²⁷⁵ - Roles are cleaned up when an associated repository has been deleted

UI

- [\[NEXUS-16387\]](#)²⁷⁶ - Rebuild of browse nodes is only performed on available repositories
- [\[NEXUS-16584\]](#)²⁷⁷ - Fix to uploading large artifacts

Maven

- [\[NEXUS-16393\]](#)²⁷⁸ - Correctly merge non-timestamped maven-metadata.xml files

²⁷³ <https://issues.sonatype.org/browse/NEXUS-12331>

²⁷⁴ <https://issues.sonatype.org/browse/NEXUS-11446>

²⁷⁵ <https://issues.sonatype.org/browse/NEXUS-16227>

²⁷⁶ <https://issues.sonatype.org/browse/NEXUS-16387>

²⁷⁷ <https://issues.sonatype.org/browse/NEXUS-16584>

- [NEXUS-16539^{279]}] - 401 responses now engage auto-blocking

Docker

- [NEXUS-16753^{280]}] - Connection pool leak when docker hub proxy repository receives 401 responses from auth.docker.io
- [NEXUS-16757^{281]}] - Ensures deletion of incomplete upload task

HA-C

- [NEXUS-16561^{282]}] - Some database backups were prevented

Repository Manager 3.10.0

4/5/2018

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.10.0. A summary of the highlights in this release is shown below.

See the [complete release notes²⁸³](#) for all resolved issues.

New and Noteworthy

Component Tagging and Custom Attributes

This release includes a preview of our REST endpoints for component tagging (NXRM Pro only), which allows NXRM users to search for components and tag them, creating arbitrary collections of components. Tags also support custom attributes, which makes it possible to attach user-supplied information to tagged components.

In an upcoming release, it will be possible to tag components as they are uploaded to NXRM, making it possible to identify collections of components as a 'build'. This will form the basis of our upcoming staging features.

For more information please review the [tagging\(see page 357\)](#) documentation.

²⁷⁹ <https://issues.sonatype.org/browse/NEXUS-16539>

²⁸⁰ <https://issues.sonatype.org/browse/NEXUS-16753>

²⁸¹ <https://issues.sonatype.org/browse/NEXUS-16757>

²⁸² <https://issues.sonatype.org/browse/NEXUS-16561>

²⁸³ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17453>

Hosted NuGet Queries Now Return Supported Frameworks

[NEXUS-14839](#)²⁸⁴

Hosted NuGet queries will now return supported frameworks that don't have listed dependencies - previously frameworks without dependencies were incorrectly ignored.

This fix applies to all new packages that are deployed. If you have existing packages that are affected and can't redeploy them this script ([NEXUS-14839-fixNugetDependencies.groovy](#)²⁸⁵) will need to be run once to successful completion on version 3.10.0 and greater.

Docker Push of Multilayer Images Now Works in HA-C

[NEXUS-15722](#)²⁸⁶

Docker push of images containing multiple layers to an NXRM HA cluster running behind a load balancer is now properly handled.

General Improvements

LDAP

- [\[NEXUS-15816\]](#)²⁸⁷ - Paged results sets can now be disabled in LDAP searches

NuGet

- [\[NEXUS-10030\]](#)²⁸⁸ - Pre-released NuGet packages are now identified by their version string to workaround a NuGet bug

REST

- [\[NEXUS-16425\]](#)²⁸⁹ - Download endpoint now only returns the jar file if Maven classifier parameter is set

Security,UI

- [\[NEXUS-16248\]](#)²⁹⁰ - Roles with circular references can no longer be created

²⁸⁴ <https://issues.sonatype.org/browse/NEXUS-14839>

²⁸⁵ https://issues.sonatype.org/secure/attachment/80034/80034_NEXUS-14839-fixNugetDependencies.groovy

²⁸⁶ <https://issues.sonatype.org/browse/NEXUS-15722>

²⁸⁷ <https://issues.sonatype.org/browse/NEXUS-15816>

²⁸⁸ <https://issues.sonatype.org/browse/NEXUS-10030>

²⁸⁹ <https://issues.sonatype.org/browse/NEXUS-16425>

²⁹⁰ <https://issues.sonatype.org/browse/NEXUS-16248>

Tree View

- [NEXUS-16470²⁹¹] - User-supplied filters are now properly escaped and sanitized

Upload UI

- [NEXUS-16454²⁹²] - Raw repository upload now works in IE11
- [NEXUS-16503²⁹³] - Artifacts can now be uploaded to the root of a Raw repository

Yum

- [NEXUS-15745²⁹⁴] - Yum proxy is now able to remove absolute URLs for metadata files that aren't at the root of a repository

Repository Manager 3.9.0

2/28/2018

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.9.0. A summary of the highlights in this release is shown below.

See the [complete release notes²⁹⁵](#) for all resolved issues.

New and Noteworthy

Upload components to a repository from the UI

NEXUS-10121²⁹⁶

This is the new and improved version of the upload feature that exists in Nexus Repository 2. For Nexus Repository 3, we support uploads to hosted Maven, Raw, npm, PyPI, NuGet, and RubyGems repositories.

Nexus Firewall now supported on OSS

NEXUS-16155²⁹⁷

²⁹¹ <https://issues.sonatype.org/browse/NEXUS-16470>

²⁹² <https://issues.sonatype.org/browse/NEXUS-16454>

²⁹³ <https://issues.sonatype.org/browse/NEXUS-16503>

²⁹⁴ <https://issues.sonatype.org/browse/NEXUS-15745>

²⁹⁵ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17425>

²⁹⁶ <https://issues.sonatype.org/browse/NEXUS-10121>

²⁹⁷ <https://issues.sonatype.org/browse/NEXUS-16155>

This release makes it possible to use [Nexus Firewall with Nexus Repository OSS](#)²⁹⁸, for those who want the ability to block bad components from entering their repositories, but don't necessarily need the full set of capabilities in Nexus Repository Pro.

Yum Proxy and Hosted support conditional GET

[NEXUS-15815](#)²⁹⁹, [NEXUS-16066](#)³⁰⁰

When making request to either a hosted or proxy yum repository, Nexus will respond properly when a *If-Modified-Since* header is present.

Remove support for the non-gzipped specs 4.8 from Rubygems

[NEXUS-14885](#)³⁰¹

The public RubyGems repository has removed support for the uncompressed specs.4.8 index file and this ticket removes it from NXRM.

Anyone running a Rubygems client earlier than 1.8 will have to update when upgrading to the latest version of NXRM.

If you have any third party tools that are accessing the specs.4.8 endpoint directly they will receive a 404. They should be redirected to the specs.4.8.gz endpoint instead.

Example old endpoint = <http://localhost:8080/repository/ruby-hosted/specs.4.8>

Example new endpoint = <http://localhost:8080/repository/ruby-hosted/specs.4.8.gz>

General Improvements

NPM

- [NEXUS-10255]³⁰²] NPM proxy failed with 404 for requests with version specified

Repository Health Check, Upgrade

- [NEXUS-15746]³⁰³] Health check config database upgrade sometimes fails

²⁹⁸ <https://help.sonatype.com/learning/iq-server-lifecycle-and-firewall/firewall-quick-start-guide>

²⁹⁹ <https://issues.sonatype.org/browse/NEXUS-15815>

³⁰⁰ <https://issues.sonatype.org/browse/NEXUS-16066>

³⁰¹ <https://issues.sonatype.org/browse/NEXUS-14885>

³⁰² <https://issues.sonatype.org/browse/NEXUS-10255>

³⁰³ <https://issues.sonatype.org/browse/NEXUS-15746>

Yum

- Yum hosted caches 404 responses for files unnecessarily due to negative cache [[NEXUS-15795](#)³⁰⁴]
handler

Tasks

- [[NEXUS-15461](#)³⁰⁵] Allow more tasks to be canceled

Repository Manager 3.8.0

02/05/2018

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.8.0. A summary of the highlights in this release is shown below.

For more detail see the [complete release notes](#)³⁰⁶.

Multiple XSS Vulnerabilities

Multiple XSS vulnerabilities have been discovered in Nexus Repository 3.x up to and including version 3.7.1.

We recommend upgrading to 3.8.0 or later immediately. See our [support knowledge base article](#)³⁰⁷ for more information.

Yum Hosted

[NEXUS-10191](#)³⁰⁸

With our initial support for Yum Proxy released in version 3.5.0 we are now continuing on with the Yum Hosted. This new feature is no longer built on top of Maven and no longer dependant on the external createrepo program. Yum hosting is now platform independent. Yum group repository and support for upgrading 2.x yum repositories to 3.x will be included in future releases.

Use *permissive* Deploy Policy if you're using Maven to deploy RPMs to Yum Hosted.

<https://issues.sonatype.org/browse/NEXUS-15795>³⁰⁴

³⁰⁵ <https://issues.sonatype.org/browse/NEXUS-15461>

³⁰⁶ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17364>

³⁰⁷ <https://support.sonatype.com/hc/en-us/articles/360000134968>

³⁰⁸ <https://issues.sonatype.org/browse/NEXUS-10191>

REST API deprecating /siesta

[NEXUS-14940](#)³⁰⁹

We have removed "/siesta/" from all of our REST endpoints, so you'll need to update your integrations. For example, the "/service/siesta/rest/v1/script" endpoint has been moved to "/service/rest/v1/script".

Upgrading from 3.x

This version upgrades Eclipse Jetty from 9.3.x to 9.4.x³¹⁰. This upgrade required a line to be removed from the shipped <install-dir>/etc/jetty/jetty-http.xml and <install-dir>/etc/jetty/jetty-https.xml as compared to previous versions.

Startup will fail if you try to use a jetty configuration file from a previous version that contains the following line:

line that will fail startup if present in jetty-http.xml or jetty-https.xml

```
<Set name="selectorPriorityDelta"><Property name="jetty.http.selectorPriorityDelta" default="0"/></Set>
```

This highlights why it is important to always compare install files you previously modified on upgrade as recommended by [our upgrade instructions](#)³¹¹.

Upgrading from 2.x

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.6. See the [upgrade compatibility matrix](#)(see page 146) for more information.

General Improvements

Blobstore,UI

- [\[NEXUS-15467\]](#)³¹² - Make blob store type field not editable

Bootstrap

- [\[NEXUS-14956\]](#)³¹³ - Upgrade to Eclipse Jetty 9.4.x

³⁰⁹ <https://issues.sonatype.org/browse/NEXUS-14940>

³¹⁰ <https://issues.sonatype.org/browse/NEXUS-14956>

³¹¹ <https://support.sonatype.com/hc/en-us/articles/115000350007>

³¹² <https://issues.sonatype.org/browse/NEXUS-15467>

³¹³ <https://issues.sonatype.org/browse/NEXUS-14956>

Bower,Security

- [NEXUS-12452³¹⁴] - Bower install no longer fails when user has only group level privileges

Content Selectors,Tree View

- [NEXUS-15545³¹⁵] - Tree view now works properly with content selectors

Fabric

- [NEXUS-14969³¹⁶] - HA-C nodes now properly rejoin their cluster after cluster shutdown
- [NEXUS-15084³¹⁷] - HA-C properly syncs user accounts between nodes

LDAP

- [NEXUS-15147³¹⁸] - Prevent ConcurrentModificationException when editing multiple user roles

Logging

- [NEXUS-15364³¹⁹] - Logging from different task threads may log to the same task log if tasks are started within the same second

Maven

- [NEXUS-12482³²⁰] - Inconsistent behaviour with upload to snapshot repository fixed

NPM

- [NEXUS-15282³²¹] - NPM allows redeploys despite Deploy Policy
- [NEXUS-15425³²²] - Assets now properly updated when a npm package is republished

³¹⁴ <https://issues.sonatype.org/browse/NEXUS-12452>

³¹⁵ <https://issues.sonatype.org/browse/NEXUS-15545>

³¹⁶ <https://issues.sonatype.org/browse/NEXUS-14969>

³¹⁷ <https://issues.sonatype.org/browse/NEXUS-15084>

³¹⁸ <https://issues.sonatype.org/browse/NEXUS-15147>

³¹⁹ <https://issues.sonatype.org/browse/NEXUS-15364>

³²⁰ <https://issues.sonatype.org/browse/NEXUS-12482>

³²¹ <https://issues.sonatype.org/browse/NEXUS-15282>

³²² <https://issues.sonatype.org/browse/NEXUS-15425>

Outreach

- [NEXUS-15466³²³] - Welcome screen content is now displayed for administrators who are mapped in via LDAP group

REST

- [NEXUS-15202³²⁴] - Take classifier into account when downloading a jar through the REST endpoint /rest/beta/search/assets/download
- [NEXUS-15088³²⁵] - Incorrect error response code 406 for bad ID in DELETE /component
- [NEXUS-15089³²⁶] - Error response code 204 not listed in REST API codes for component and asset delete

Yum

- [NEXUS-15131³²⁷] - Component naming for Yum Proxy now matches RPM header

2017 Release Notes

Repository Manager 3.7.1

12/28/2017

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.7.1. A summary of the highlights in this release is shown below.

For more detail see the [complete release notes³²⁸](#).

Offline Repositories for Tree/HTML views [NEXUS-15278³²⁹]

This release fixes an issue where some content may be missing in the tree or html views for offline repositories. We highly recommend users of 3.7.0 upgrade to this patch release.

³²³ <https://issues.sonatype.org/browse/NEXUS-15466>

³²⁴ <https://issues.sonatype.org/browse/NEXUS-15202>

³²⁵ <https://issues.sonatype.org/browse/NEXUS-15088>

³²⁶ <https://issues.sonatype.org/browse/NEXUS-15089>

³²⁷ <https://issues.sonatype.org/browse/NEXUS-15131>

³²⁸ <https://issues.sonatype.org/secure/ReleaseNote.jspa?version=17368&styleName=Html&projectId=10001>

³²⁹ <https://issues.sonatype.org/browse/NEXUS-15278>

Repository Manager 3.7.0

12/19/2017

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.7.0. A summary of the highlights in this release is shown below.

For more detail see the [complete release notes](#)³³⁰.

Upgrade to 3.7.1 Recommended

If you upgrade to 3.7.0 from a version that has a repository marked offline, then the new [Tree View](#)³³¹ feature introduced in 3.7.0 may fail to render all repository content. This [known issue](#)³³² does not cause builds to fail or the search user interface from working, only tree view is affected. This issue is resolved in 3.7.1 and newer.

Tree and HTML views replace prior Browse features

[NEXUS-12531](#)³³³, [NEXUS-12518](#)³³⁴

The browse experience has been upgraded to include a [Tree View](#)(see page 227) and HTML View to help you find your assets faster and easier. During this process, we also exhaustively tested for scale and upgrade across all formats. The search experience remains the same as prior versions.

When upgrading, Nexus Repository Manager will automatically begin creating the tree node data. Depending on hardware, this data should take ~30 minutes to process 3 million assets and the storage of the component database should increase by ~30%. While the assets are being processed browse will display a partial tree.

NPM command line Search restored

[NEXUS-13151](#)³³⁵

npm deprecated the /all/ endpoint to their search interface earlier this year. This improvement restores npm search via that interface by enabling the replacement /v1/ endpoint. We took this opportunity to add some scalability and performance tweaks as well.

³³⁰ <https://issues.sonatype.org/secure/ReleaseNote.jspa?version=17228&styleName=Html&projectId=10001>

³³¹ <https://help.sonatype.com/display/NXRM3M/Browsing+Repositories+and+Repository+Groups>

³³² <https://issues.sonatype.org/browse/NEXUS-15278>

³³³ <https://issues.sonatype.org/browse/NEXUS-12531>

³³⁴ <https://issues.sonatype.org/browse/NEXUS-12518>

³³⁵ <https://issues.sonatype.org/browse/NEXUS-13151>

Upgrading from 2.x

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.5. See the [upgrade compatibility matrix](#)(see page 146) for more information.

General Highlights

Docker

- [\[NEXUS-13894\]](#)³³⁶ - Docker unauthenticated access shows "unknown: unknown" console output
- [\[NEXUS-12216\]](#)³³⁷ - Support pushing Docker Windows Container images and loosen manifest validation to allow for 'foreign-layers'

Migration

- [\[NEXUS-10162\]](#)³³⁸ - migrating NXRM2 to NXRM3 automatically enables legacy content URLs
- [\[NEXUS-13329\]](#)³³⁹ - Maven artifact whose case does not match the version folder they reside in are not migrated

Repository Health Check

- [\[NEXUS-14960\]](#)³⁴⁰ - healthcheck.properties can contain references to repositories that do not exist causing 3.6.1 upgrade to fail

Repository Manager 3.6.2

11/28/2017

This release is a bug fix release for a critical bug that affects LDAP authentication in HA configurations.

See the [complete release notes](#)³⁴¹.

³³⁶ <https://issues.sonatype.org/browse/NEXUS-13894>

³³⁷ <https://issues.sonatype.org/browse/NEXUS-12216>

³³⁸ <https://issues.sonatype.org/browse/NEXUS-10162>

³³⁹ <https://issues.sonatype.org/browse/NEXUS-13329>

³⁴⁰ <https://issues.sonatype.org/browse/NEXUS-14960>

³⁴¹ <https://issues.sonatype.org/secure/ReleaseNote.jspa?version=17355&projectId=10001>

Upgrading from 2.x

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.5. See the upgrade compatibility matrix for more information.

Repository Manager 3.6.1

11/15/2017

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.6.1.

See the [complete release notes](#)³⁴² for all resolved issues.

New and Noteworthy

This release is a rollup of 70+ bug fixes and general improvements we've made over the last couple of months.

Upgrading from 2.x

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.5. See the upgrade compatibility matrix for more information.

General Improvements

Blobstore

- [\[NEXUS-14707\]](#)³⁴³ - Additional check for blobstore default integrity strategy

Docker

- [\[NEXUS-14488\]](#)³⁴⁴ - Cannot perform docker pull against some Docker-Hub proxy
- [\[NEXUS-14512\]](#)³⁴⁵ - add anonymous search for docker repositories

³⁴² <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17342>

³⁴³ <https://issues.sonatype.org/browse/NEXUS-14707>

³⁴⁴ <https://issues.sonatype.org/browse/NEXUS-14488>

³⁴⁵ <https://issues.sonatype.org/browse/NEXUS-14512>

Maven

- [\[NEXUS-13949\]³⁴⁶](https://issues.sonatype.org/browse/NEXUS-13949) - Remove snapshots from Maven repository remove if released option may progress slowly

NuGet, Raw

- [\[NEXUS-11962\]³⁴⁷](https://issues.sonatype.org/browse/NEXUS-11962) - add legacy URL support for non-maven repositories

NPM

- [\[NEXUS-13207\]³⁴⁸](https://issues.sonatype.org/browse/NEXUS-13207) - NPM group search document is not invalidated when member search documents change

Performance

- [\[NEXUS-14843\]³⁴⁹](https://issues.sonatype.org/browse/NEXUS-14843) - cursor prefetch limit set to high by default for some queries potentially leading to excessive memory usage

REST API

- [\[NEXUS-14730\]³⁵⁰](https://issues.sonatype.org/browse/NEXUS-14730) - expose REST API developer documentation inside the user interface
- [\[NEXUS-14603\]³⁵¹](https://issues.sonatype.org/browse/NEXUS-14603) - REST Asset Search JSON API

Security

- [\[NEXUS-10896\]³⁵²](https://issues.sonatype.org/browse/NEXUS-10896) - record and expose the authenticated user who uploaded a component to a hosted repository
- [\[NEXUS-14515\]³⁵³](https://issues.sonatype.org/browse/NEXUS-14515) - Display the IP address of the user who uploaded a component to a hosted repository

Upgrade

- [\[NEXUS-12907\]³⁵⁴](https://issues.sonatype.org/browse/NEXUS-12907) - Upgrade from 2.x to 3.x Hangs on Group Repos containing only Staging Repos

³⁴⁶ <https://issues.sonatype.org/browse/NEXUS-13949>

³⁴⁷ <https://issues.sonatype.org/browse/NEXUS-11962>

³⁴⁸ <https://issues.sonatype.org/browse/NEXUS-13207>

³⁴⁹ <https://issues.sonatype.org/browse/NEXUS-14843>

³⁵⁰ <https://issues.sonatype.org/browse/NEXUS-14730>

³⁵¹ <https://issues.sonatype.org/browse/NEXUS-14603>

³⁵² <https://issues.sonatype.org/browse/NEXUS-10896>

³⁵³ <https://issues.sonatype.org/browse/NEXUS-14515>

³⁵⁴ <https://issues.sonatype.org/browse/NEXUS-12907>

YUM

- [NEXUS-14058³⁵⁵] - configurable repodata depth for yum proxy repos
- [NEXUS-14484³⁵⁶] - Add support for proxying yum repositories that have sha1 checksums

Repository Manager 3.6.0

09/27/17

Sonatype is pleased to announce the immediate availability of Nexus Repository 3.6.0 OSS and Pro. A summary of the highlights in this release is shown below.

Please see [JIRA for the complete list of changes](#)³⁵⁷.

Docker Anonymous Support

[NEXUS-10813](#)³⁵⁸

This change makes it possible for [anonymous users to be granted read-only \(pull\) access](#)³⁵⁹. It's enabled by a new Docker repository connector option and realm. Anonymous read support can be deactivated by default and must be enabled on each Docker repository individually. Ultimately, this is useful as it allows users to consume and share Docker images with no credentials needed.

Goodbye JEXL, Hello CSEL

[NEXUS-13942](#)³⁶⁰

We've discovered that Apache JEXL based content selectors aren't fast enough for our upcoming tree view feature, which does many more evaluations than other browsing methods. To address this we're introducing our own [Content Selector Expression Language](#)(see page 0) (CSEL) to support current features, such as [Upgrading](#)(see page 193), and future improvements that rely on optimal performance.

Nexus Repository will automatically upgrade as many of your existing JEXL selectors to CSEL selectors as possible. Any remaining JEXL selectors will continue to function but with this new expression language in place, future releases of the repository manager will be more performant.

³⁵⁵ <https://issues.sonatype.org/browse/NEXUS-14058>

³⁵⁶ <https://issues.sonatype.org/browse/NEXUS-14484>

³⁵⁷ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17147>

³⁵⁸ <https://issues.sonatype.org/browse/NEXUS-10813>

³⁵⁹ <https://help.sonatype.com/display/NXRM3/Private+Registry+for+Docker#PrivateRegistryforDocker-AnonymousReadAccess>

³⁶⁰ <https://issues.sonatype.org/browse/NEXUS-13942>

Provisioning API and Group Repositories

[NEXUS-14038](#)³⁶¹

Groups repositories created through the [provisioning API](#)(see page 464) will now preserve the order of their members. Previously, the initial ordering was set correctly, but jumbled on server restart.

-  For existing repositories affected by this, the workaround is to load the group repository in the UI, reorder the members as desired, and save.

Firewall and IQ Server

Firewall now requires the [IQ Server connection capability to be configured and enabled](#)³⁶². Previously, it would process transactions (e.g. audit) regardless, but a recent change now requires this capability.

Upgrading from 2.x

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.5. See the [upgrade compatibility matrix](#)(see page 146) for more information.

Repository Manager 3.5.2

09/12/17

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.5.2.

See the complete [release notes](#)³⁶³ for all resolved issues.

New and Noteworthy

This is a targeted release to address excessive thread creation associated with task progress logging.

The vast majority of customers will benefit from updating to this release.

³⁶¹ <https://issues.sonatype.org/browse/NEXUS-14038>

³⁶²<https://help.sonatype.com/display/NXI/Integrating+NXRM+3.x+and+IQ+Server#IntegratingNXRM3.xandIQServer-ConnectingtoIQServer>

³⁶³ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17224>

General Improvements

Task Logging

- [NEXUS-14227³⁶⁴] Bug Thread count increases linearly with scheduled task execution

Repository Manager 3.5.1

08/18/17

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.5.1.

See the [complete release notes](#)³⁶⁵ for all resolved issues.

New and Noteworthy

This is a targetted release for a select set of Pro licensed customers aimed at resolving some critical database concurrency issues.

The vast majority of customers will observe no immediate benefit from updating to this release.

Upgrading from 2.x

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.5. See the upgrade compatibility matrix for more information.

General Improvements

Database

- [NEXUS-14087³⁶⁶] **Improvement** upgrade OrientDB dependency to version 2.2.26

Repository Manager 3.5.0

08/03/17

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.5.

³⁶⁴ <https://issues.sonatype.org/browse/NEXUS-14227>

³⁶⁵ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17153>

³⁶⁶ <https://issues.sonatype.org/browse/NEXUS-14087>

See the [complete release notes](#)³⁶⁷ for all resolved issues.

New and Noteworthy

Yum Proxy Repository Support

You can now define [Yum proxy repositories](#)(see page 460). In Nexus Repository 2, Yum support was built on top of maven repositories, this time around, we're building yum as a first-class format.

Yum hosted and group repository support will be in a yet to be announced future release and this time around we're building Yum support to be platform independent such that it will not have a dependency on the external createrepo program.

Note that upgrading Nexus 2.x Maven 2 format Yum enabled proxy repositories is not supported - your new Nexus 3.x Yum proxy repository will download remote rpm files and metadata as needed. Give it a try and let us know what you think!

Upgrade Support for Firewall Enabled Repositories

For Firewall customers looking to upgrade their Nexus Repository instances from 2.x to 3.x, we've enhanced the upgrade wizard with support for quarantine-enabled repositories. This means that when you upgrade a 2.x instance to 3.x, the audit and quarantine history is upgraded along with it.

Upgrading to Nexus Repository 3.5.0 from 2.x requires the [just-released Nexus Repository 2.14.5](#)³⁶⁸. [IQ Server 1.33](#)³⁶⁹ is also a required minimum for upgrading Firewall-enabled repositories.

Upgrading from 2.x

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.5. See the [upgrade compatibility matrix](#)³⁷⁰ for more information.

Changes to Startup Files

A JVM optimization was making debugging some of your reported issues difficult; we have corrected this by changing the default JVM options. [[NEXUS-13777](#)³⁷¹]

Picking up this change will require the following line in file `bin/nexus.vmoptions` (now the default):

`-XX:-OmitStackTraceInFastThrow`

Per-Task Log Files

³⁶⁷ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=17139>

³⁶⁸ <https://sonatype.zendesk.com/hc/en-us/articles/115010637927>

³⁶⁹ <https://support.sonatype.com/hc/en-us/articles/213463638>

³⁷⁰ <https://support.sonatype.com/hc/en-us/articles/115007568788>

³⁷¹ <https://issues.sonatype.org/browse/NEXUS-13777>

Scheduled tasks will now [output to their own log files](#)³⁷² allowing for cleaner separation of task log output from regular log output. Task logs are retained up to 30 days and can be found near the existing log files under sonatype-work/nexus3/log/tasks. [[NEXUS-13352](#)³⁷³]

General Improvements

Backup

- [[NEXUS-13486](#)³⁷⁴] Improvement prevent restoring database backups with mismatched versions

Blobstore

- [[NEXUS-12389](#)³⁷⁵] Improvement if the component database references a soft-deleted blob then prevent blob store compaction task from hard deleting the blob

Bootstrap

- [[NEXUS-13485](#)³⁷⁶] Improvement warn in UI when ulimit < 65536 on Linux or OSX
- [[NEXUS-11870](#)³⁷⁷] Improvement jetty-https.xml obfuscated keystore truststore password values are confusing

CLM, Upgrade

- [[NEXUS-13901](#)³⁷⁸] Improvement support upgrading Nexus Repository Manager 2 Firewall enabled repositories to Nexus Repository Manager 3

Database

- [[NEXUS-13304](#)³⁷⁹] Bug 500 responses from Nexus after enabling quarantine on NuGet proxy repository

Docker

- [[NEXUS-13363](#)³⁸⁰] Bug Conditional GET requests for Docker image layers always download the layer when proxying another Nexus performance

Logging

- [[NEXUS-12968](#)³⁸¹] Bug Uninformative log message in ProxyFacetSupport - Content not present for throwing exception

³⁷² <https://help.sonatype.com/display/HSCM/Configuration+-+NXRM+3#Configuration-NXRM3-TaskLogging>

³⁷³ <https://issues.sonatype.org/browse/NEXUS-13352>

³⁷⁴ <https://issues.sonatype.org/browse/NEXUS-13486>

³⁷⁵ <https://issues.sonatype.org/browse/NEXUS-12389>

³⁷⁶ <https://issues.sonatype.org/browse/NEXUS-13485>

³⁷⁷ <https://issues.sonatype.org/browse/NEXUS-11870>

³⁷⁸ <https://issues.sonatype.org/browse/NEXUS-13901>

³⁷⁹ <https://issues.sonatype.org/browse/NEXUS-13304>

³⁸⁰ <https://issues.sonatype.org/browse/NEXUS-13363>

³⁸¹ <https://issues.sonatype.org/browse/NEXUS-12968>

- [NEXUS-13777³⁸²] Bug JVM optimizations may log exceptions without stack traces by default

Migration, NuGet

- [NEXUS-13554³⁸³] Bug A NuGet package that is in Nexus 2.x storage but not in its database causes a NullPointerException on migration to Nexus 3

NPM

- [NEXUS-12457³⁸⁴] Bug npm proxy receiving connection reset responds to client with status 500 instead of 404

NuGet

- [NEXUS-12339³⁸⁵] Bug Faulty result ordering for NuGet searches
- [NEXUS-10144³⁸⁶] Improvement improve robustness of NuGet case insensitive package ID matching

Repository

- [NEXUS-10243³⁸⁷] Bug Content type exception uploading tar files - identified as application-gtar instead of application/tar

UI

- [NEXUS-9546³⁸⁸] Improvement add better validation on task configuration numeric fields

Yum

- [NEXUS-13900³⁸⁹] Improvement add YUM proxy repository support
-

Repository Manager 3.4.0

01/18/17

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.4.

See the [complete release notes³⁹⁰](#) for all resolved issues.

New and Noteworthy

³⁸² <https://issues.sonatype.org/browse/NEXUS-13777>

³⁸³ <https://issues.sonatype.org/browse/NEXUS-13554>

³⁸⁴ <https://issues.sonatype.org/browse/NEXUS-12457>

³⁸⁵ <https://issues.sonatype.org/browse/NEXUS-12339>

³⁸⁶ <https://issues.sonatype.org/browse/NEXUS-10144>

³⁸⁷ <https://issues.sonatype.org/browse/NEXUS-10243>

³⁸⁸ <https://issues.sonatype.org/browse/NEXUS-9546>

³⁸⁹ <https://issues.sonatype.org/browse/NEXUS-13900>

³⁹⁰ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=16825>

Upgrading from 2.x

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.4. See the [upgrade compatibility matrix](#)³⁹¹ for more information.

Updated System Requirements: Increase Default File Handle limits

On Linux and Mac OSX installs, the default process file handle limits available may be too small and may cause data corruption or other significant problems if not explicitly increased. We recommend all deployments (even previous versions) raise their available file descriptors to 65536 immediately. How to implement this change is summarized in [the system requirements](#)³⁹². [[NEXUS-12041](#)³⁹³].

General Performance/Scalability Improvements

All sorts of performance optimizations for existing features are in this release, primarily with scheduled tasks, browse and search. Issues which were causing OutOfMemory errors were also squashed. We continue to schedule fixing any remaining severe issues with extreme prejudice and we are confident this is the best performing 3.x release to date.

General Improvements

Bootstrap

- [[NEXUS-13098](#)³⁹⁴] Improvement log and prevent startup when the started Nexus version is using a data directory from a newer Nexus version
- [[NEXUS-12041](#)³⁹⁵] Bug WARN org.elasticsearch.env max file descriptors for elasticsearch process likely too low, consider increasing logged at start

Bower,PyPi

- [[NEXUS-13137](#)³⁹⁶] Improvement Purge unused components and assets task should support PyPi and Bower proxy repositories

Build

- [[NEXUS-12540](#)³⁹⁷] Bug Unable to Start Nexus-public
- [[NEXUS-12397](#)³⁹⁸] Bug nexus-public base template binary fails to start due to DependencyResolver\$UnresolvedDependencyException

Crowd

³⁹¹ <https://support.sonatype.com/hc/en-us/articles/115007568788>

³⁹² <https://sonatype.zendesk.com/hc/en-us/articles/115006448847#filehandles>

³⁹³ <https://issues.sonatype.org/browse/NEXUS-12041>

³⁹⁴ <https://issues.sonatype.org/browse/NEXUS-13098>

³⁹⁵ <https://issues.sonatype.org/browse/NEXUS-12041>

³⁹⁶ <https://issues.sonatype.org/browse/NEXUS-13137>

³⁹⁷ <https://issues.sonatype.org/browse/NEXUS-12540>

³⁹⁸ <https://issues.sonatype.org/browse/NEXUS-12397>

- [NEXUS-12405³⁹⁹] Bug Crowd realm is missing 'Clear Cache' option like the LDAP realm has

Database

- [NEXUS-13431⁴⁰⁰] Improvement upgrade OrientDB to version 2.2.21

Docker

- [NEXUS-12711⁴⁰¹] Bug Nexus docker registry delete REST api partially deletes an image
- [NEXUS-13385⁴⁰²] Bug java.util.ConcurrentModificationException possible with Docker UploadManager during POST to blobs/uploads

Documentation

- [NEXUS-13000⁴⁰³] Bug clarify file name references in documentation for upgrading from 3.2 to 3.3

LDAP,UI

- [NEXUS-13071⁴⁰⁴] Bug Unfiltered LDAP user search will retrieve all users from an LDAP server, which can result in an OOM

Logging

- [NEXUS-12908⁴⁰⁵] Bug Connection issue to IQ server is only logged at DEBUG level
- [NEXUS-13141⁴⁰⁶] Bug when merging maven-metadata.xml for a group request fails the repository ID containing the bad metadata is not logged
- [NEXUS-13096⁴⁰⁷] Bug com.orientechnologies.common.profiler.OAbstractProfiler\$MemoryChecker log spam every 2 minutes
- [NEXUS-13371⁴⁰⁸] Bug BlobAttributes deletedReason is logged as reason: null instead of the actual reason

Maven Repository

- [NEXUS-12844⁴⁰⁹] Improvement Upgrade Apache Tika dependency to 1.14

Maven2

399 <https://issues.sonatype.org/browse/NEXUS-12405>

400 <https://issues.sonatype.org/browse/NEXUS-13431>

401 <https://issues.sonatype.org/browse/NEXUS-12711>

402 <https://issues.sonatype.org/browse/NEXUS-13385>

403 <https://issues.sonatype.org/browse/NEXUS-13000>

404 <https://issues.sonatype.org/browse/NEXUS-13071>

405 <https://issues.sonatype.org/browse/NEXUS-12908>

406 <https://issues.sonatype.org/browse/NEXUS-13141>

407 <https://issues.sonatype.org/browse/NEXUS-13096>

408 <https://issues.sonatype.org/browse/NEXUS-13371>

409 <https://issues.sonatype.org/browse/NEXUS-12844>

- [NEXUS-13085⁴¹⁰] Bug IllegalArgumentException Version mismatch may be logged when GA maven-metadata.xml versions are merged in a group repository request

NPM

- [NEXUS-13168⁴¹¹] Bug NullPointerException on npm search when invalidating cache
- [NEXUS-11910⁴¹²] Bug npm search against group repository fails with HTTP 500 due to not properly supporting /-/all resource
- [NEXUS-12869⁴¹³] Bug npm publish a large package may cause java.lang.OutOfMemoryError: Java heap space when parsing the JSON payload performance
- [NEXUS-12304⁴¹⁴] Bug `npm publish` on an already published package version does not update all changed package metadata
- [NEXUS-12716⁴¹⁵] Bug NullPointerException when running npm search

NuGet

- [NEXUS-12064⁴¹⁶] Bug JPQLGenerator.toJpqlLiteral NullPointerException for NuGet /Packages() resource as submitted by OctopusDeploy

Proxy Repository

- [NEXUS-13378⁴¹⁷] Improvement Limit multiple outbound upstream requests for the same proxied asset performance

Repository

- [NEXUS-11215⁴¹⁸] Bug valid .woff files fail Strict Content Type Validation with 400 response

Repository Health Check

- [NEXUS-13087⁴¹⁹] Bug caching asset download counts under high unique request volume can lead to OutOfMemoryError and instability performance
- [NEXUS-13432⁴²⁰] Bug asset download count feature contributes log noise and heap memory spike every 24 hrs when deleting old download counts performance

410 <https://issues.sonatype.org/browse/NEXUS-13085>

411 <https://issues.sonatype.org/browse/NEXUS-13168>

412 <https://issues.sonatype.org/browse/NEXUS-11910>

413 <https://issues.sonatype.org/browse/NEXUS-12869>

414 <https://issues.sonatype.org/browse/NEXUS-12304>

415 <https://issues.sonatype.org/browse/NEXUS-12716>

416 <https://issues.sonatype.org/browse/NEXUS-12064>

417 <https://issues.sonatype.org/browse/NEXUS-13378>

418 <https://issues.sonatype.org/browse/NEXUS-11215>

419 <https://issues.sonatype.org/browse/NEXUS-13087>

- [NEXUS-13026⁴²¹] Bug repository health check column displays 0 total and 0 bad even if repository has content

Repository Health Check, Security

- [NEXUS-12485⁴²²] Bug add privilege that controls access to health check summary report

RubyGems

- [NEXUS-13178⁴²³] Bug gem development dependencies are treated as runtime dependencies
- [NEXUS-12373⁴²⁴] Bug parsing dates in some gemspec files could fail with IllegalArgumentException: Invalid format

Scheduled Tasks

- [NEXUS-12780⁴²⁵] Bug task scheduler threads may deadlock at QuartzTaskJob.mayBlock() when more than 20 blocking tasks are encountered performance
- [NEXUS-13584⁴²⁶] Improvement add task that can Restore Asset/Component metadata to component database from Blob Store contents
- [NEXUS-13092⁴²⁷] Bug slow performance and metadata rebuild failures when running "Remove snapshots from Maven repository" against large datasets performance
- [NEXUS-13130⁴²⁸] Bug Timed out reading query result from queue when running purge unused snapshots task performance

Search

- [NEXUS-13163⁴²⁹] Improvement Use bulk API for incremental Elasticsearch updates performance
- [NEXUS-13466⁴³⁰] Bug SearchServiceImpl ERROR Elasticsearch index thread pool is overloaded message may appear in nexus.log performance

Transport

- [NEXUS-13136⁴³¹] Bug NullPointerException is thrown when user-agent header value is not present

UI

421 <https://issues.sonatype.org/browse/NEXUS-13026>

422 <https://issues.sonatype.org/browse/NEXUS-12485>

423 <https://issues.sonatype.org/browse/NEXUS-13178>

424 <https://issues.sonatype.org/browse/NEXUS-12373>

425 <https://issues.sonatype.org/browse/NEXUS-12780>

426 <https://issues.sonatype.org/browse/NEXUS-13584>

427 <https://issues.sonatype.org/browse/NEXUS-13092>

428 <https://issues.sonatype.org/browse/NEXUS-13130>

429 <https://issues.sonatype.org/browse/NEXUS-13163>

430 <https://issues.sonatype.org/browse/NEXUS-13466>

431 <https://issues.sonatype.org/browse/NEXUS-13136>

- [NEXUS-10419⁴³²] Bug nested webapp context path Nexus 3 breaks the UI

User Token

- [NEXUS-13127⁴³³] Bug Anonymous access to repositories does not work when "Require user tokens for repository authentication" is set
- [NEXUS-13126⁴³⁴] Bug When "Require user tokens for repository authentication" is set nexus does not send an authorization header

Webhooks

- [NEXUS-12855⁴³⁵] Bug Only one capability of type 'Webhook: Repository' can be created
-

Repository Manager 3.3.2

06/19/17

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.3.2.

See the complete release notes for all resolved issues.

New and Noteworthy

Browse Performance

Browsing large repositories especially with a UI text filter added, could take longer than the UI timeout of 60 seconds to complete. We have optimized internal queries to mitigate this effect. [NEXUS-13095⁴³⁶]

Search Performance

There have been multiple reports of slow or incomplete asset and component search. Under some circumstances, assets recently published or proxied into a repository would not show up in search results (despite being visible in browse and retrieval by client tooling). This behavior was due to the default indexing configuration which silently ignored over-capacity indexing requests. We've switched to a bulk indexing mode, which avoids this. [NEXUS-12520⁴³⁷]

NuGet Client Query Performance

Under some circumstances, NuGet ODATA searches could take unacceptably long to complete. Performance has been improved significantly through query optimization. [NEXUS-12983⁴³⁸]

Upgrading from 2.x

⁴³² <https://issues.sonatype.org/browse/NEXUS-10419>

⁴³³ <https://issues.sonatype.org/browse/NEXUS-13127>

⁴³⁴ <https://issues.sonatype.org/browse/NEXUS-13126>

⁴³⁵ <https://issues.sonatype.org/browse/NEXUS-12855>

⁴³⁶ <https://issues.sonatype.org/browse/NEXUS-13095>

⁴³⁷ <https://issues.sonatype.org/browse/NEXUS-12520>

⁴³⁸ <https://issues.sonatype.org/browse/NEXUS-12983>

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.4 .

General Improvements

Browse Storage

- [NEXUS-13140⁴³⁹] Improvement when browsing assets in a group repository the asset summary UI should display the containing member repository name
- [NEXUS-13095⁴⁴⁰] Bug Browse components of large repositories fails with IllegalStateException Timed out reading query result from queue performance

NuGet

- [NEXUS-12983⁴⁴¹] Bug NuGet FindPackagesById queries may perform slowly possibly leading to general non-responsiveness performance

Search

- [NEXUS-12520⁴⁴²] Bug assets visible by browsing are not available when searching due to non-optimized elasticsearch configuration rebuilding indexes
- [NEXUS-12681⁴⁴³] Bug search assets or components UI slower to respond than expected with large datasets performance

Repository Manager 3.3.1

06/01/17

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.3.1.

See the [complete release notes⁴⁴⁴](#) for all resolved issues.

New and Noteworthy

This is a targeted release to address some key issues with blob store operations.

Refetch Proxy Repository Content When a Blob is Missing

Prior to this release, if the blob (binary content) for proxy repository asset was marked as soft-deleted, an error would be logged return 500 status returned to the calling client. Now, the content will be re-fetched from the remote and if still not available, an appropriate 404 status will be returned. [NEXUS-12388⁴⁴⁵].

⁴³⁹ <https://issues.sonatype.org/browse/NEXUS-13140>

⁴⁴⁰ <https://issues.sonatype.org/browse/NEXUS-13095>

⁴⁴¹ <https://issues.sonatype.org/browse/NEXUS-12983>

⁴⁴² <https://issues.sonatype.org/browse/NEXUS-12520>

⁴⁴³ <https://issues.sonatype.org/browse/NEXUS-12681>

⁴⁴⁴ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=16921>

⁴⁴⁵ <https://issues.sonatype.org/browse/NEXUS-12388>

We have also made two internal improvements to blob write operations under concurrent load in order to increase resiliency. [[NEXUS-13030](https://issues.sonatype.org/browse/NEXUS-13030)⁴⁴⁶, [NEXUS-13032](https://issues.sonatype.org/browse/NEXUS-13032)⁴⁴⁷]

Record Blob Soft-Deletion Reason

Soft-deleted binary content intentionally remains on disk until the blob store is compacted. In rare cases some blobs were being referenced as soft-deleted unexpectedly. To help diagnose such a situation, the reason a blob is soft-deleted will be recorded in its accompanying .properties file. [[NEXUS-13035](https://issues.sonatype.org/browse/NEXUS-13035)⁴⁴⁸]

Upgrading from 2.x

If you're upgrading from Nexus Repository 2, you must first upgrade your installation to 2.14.4 .

General Improvements

Blob store

- [[NEXUS-13030](https://issues.sonatype.org/browse/NEXUS-13030)⁴⁴⁹] Improvement automatically retry blob creation when a UUID collision is detected
- [[NEXUS-13035](https://issues.sonatype.org/browse/NEXUS-13035)⁴⁵⁰] Improvement add a diagnostic reason for soft-deleting a blob to the blob properties file

Logging

- [[NEXUS-9872](https://issues.sonatype.org/browse/NEXUS-9872)⁴⁵¹] Improvement misconfigured docker proxy URL should log more details about critical failures at default log levels
- [[NEXUS-12793](https://issues.sonatype.org/browse/NEXUS-12793)⁴⁵²] Bug if java.lang.Error is thrown during request processing it may not be logged at default log levels

Proxy Repository

- [[NEXUS-13032](https://issues.sonatype.org/browse/NEXUS-13032)⁴⁵³] Improvement eliminate soft-deleted blobs and reduce transaction retries for identical proxy repository asset requests performance
- [[NEXUS-12388](https://issues.sonatype.org/browse/NEXUS-12388)⁴⁵⁴] Improvement attempt to refetch proxy repository content from remote when a referenced local blob is missing

Repository Manager 3.3.0

04/11/17

⁴⁴⁶ <https://issues.sonatype.org/browse/NEXUS-13030>

⁴⁴⁷ <https://issues.sonatype.org/browse/NEXUS-13032>

⁴⁴⁸ <https://issues.sonatype.org/browse/NEXUS-13035>

⁴⁴⁹ <https://issues.sonatype.org/browse/NEXUS-13030>

⁴⁵⁰ <https://issues.sonatype.org/browse/NEXUS-13035>

⁴⁵¹ <https://issues.sonatype.org/browse/NEXUS-9872>

⁴⁵² <https://issues.sonatype.org/browse/NEXUS-12793>

⁴⁵³ <https://issues.sonatype.org/browse/NEXUS-13032>

⁴⁵⁴ <https://issues.sonatype.org/browse/NEXUS-12388>

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.3.

See the [complete release notes](#)⁴⁵⁵ for all resolved issues.

New and Noteworthy

Git LFS Support

Nexus Repository is first to market with free support for [Git LFS](#)(see page 403).

Downloading or sharing large binary files, such as videos, images, audio recordings, and database files, can slow down the development process and negatively impact the performance of a DevOps tool chain. By managing these components in Nexus Repository, organizations can save time and benefit from increased availability, ease of file sharing, and the ability to better control access to Git LFS components.

Nexus Repository lets you store all of your software binaries, including Git LFS in a single location. With the introduction of Git LFS support, Nexus Repository now offers free support for eight components formats, including: Java, npm, NuGet, RubyGems, PyPI, Bower, and Docker.

Once you give our Git LFS support a try, we would love to hear how this feature works for you. Let us know at nexus-feedback@sonatype.com⁴⁵⁶.

Repository Health Check Revamped!

We've made a significant overhaul to how Repository Health Check (RHC) works. Our goal was to make it easier for repository administrators to not just understand, but to improve the health of their repositories over time.

RHC now shows the top five components in need of remediation, prioritized by the severity and impact of the vulnerability. It also provides download trends to help you understand how the health of your repositories is changing over time.

If you are using Nexus Repository and have not yet turned on the RHC feature, start today. What's the urgency? Perhaps you have not read the 2016 State of the Software Supply Chain Report that indicated 1 in 16 open source components downloads has a known security vulnerability. It's time to know what's in your repo.

Browsing and Proxy Repository Performance Issues Squashed

We continue to squash performance issues as they surface. This release includes some significant fixes that were affecting proxy repository cache throughput and Browse UI rendering. Try this new release - you should notice immediate gains in performance especially in these areas.

We are working some known issues with Search performance and we hope to have these rectified soon.

Upgrade Improvements and Requirements

If you're upgrading to 3.3 from Nexus Repository 2.x, you must first upgrade your installation to 2.14.4.

⁴⁵⁵ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=16629>

⁴⁵⁶ <mailto:nexus-feedback@sonatype.com>

We've corrected an issue where the upgrade process could fail if it encountered invalid or corrupt NuGet components in the 2.x repository.

We've also tightened the validation of proxy repository configuration to prevent blank or invalid remote repository URLs. Before upgrading to 3.3.0 from 3.2.1 or earlier, please ensure that your proxy repository remote URLs are valid.

General Improvements

Blob store

- [NEXUS-12496⁴⁵⁷] Bug FileBlobStore error handling makes it impossible to see what blob causes a runtime exception
- [NEXUS-12676⁴⁵⁸] Bug BlobStoreException should implement `toString()` to facilitate better exception messages
- [NEXUS-10540⁴⁵⁹] Bug BlobStoreManagerImpl is not thread-safe

Browse Storage

- [NEXUS-12678⁴⁶⁰] Bug Browse assets or components user interface slow to respond performance
- [NEXUS-12360⁴⁶¹] Improvement expose blob created and updated dates to avoid confusion with last updated date

Capabilities

- [NEXUS-10621⁴⁶²] Bug DefaultCapabilityRegistry is not thread-safe

Configuration, UI

- [NEXUS-12285⁴⁶³] Bug Remote Storage URL should be a required for proxy repository configuration

Crowd

- [NEXUS-12404⁴⁶⁴] Bug Crowd cache entries do not expire properly

Database

- [NEXUS-12040⁴⁶⁵] Bug Faulty handling of query timeouts in `OrientAsyncHelper.QueueConsumingIterable`

⁴⁵⁷ <https://issues.sonatype.org/browse/NEXUS-12496>

⁴⁵⁸ <https://issues.sonatype.org/browse/NEXUS-12676>

⁴⁵⁹ <https://issues.sonatype.org/browse/NEXUS-10540>

⁴⁶⁰ <https://issues.sonatype.org/browse/NEXUS-12678>

⁴⁶¹ <https://issues.sonatype.org/browse/NEXUS-12360>

⁴⁶² <https://issues.sonatype.org/browse/NEXUS-10621>

⁴⁶³ <https://issues.sonatype.org/browse/NEXUS-12285>

⁴⁶⁴ <https://issues.sonatype.org/browse/NEXUS-12404>

⁴⁶⁵ <https://issues.sonatype.org/browse/NEXUS-12040>

- [NEXUS-11972⁴⁶⁶] Bug JobStoreImpl should skip over malformed records to allow nexus to start

Documentation

- [NEXUS-12255⁴⁶⁷] Bug book mentions upgrade options that are not available

git-lfs

- [NEXUS-12644⁴⁶⁸] Improvement add support for git-lfs

LDAP

- [NEXUS-12020⁴⁶⁹] Bug LDAP cache entries do not expire properly
- [NEXUS-10533⁴⁷⁰] Bug EnterpriseLdapManager is not thread-safe
- [NEXUS-12250⁴⁷¹] Bug "Generic LDAP Server" UI configuration template should not have password attribute set by default

Logging

- [NEXUS-12242⁴⁷²] Bug repository requests to paths containing certain characters may fail with status 500 "Illegal character in path at index"
- [NEXUS-12334⁴⁷³] Bug Log spam when a user's session expires while viewing the repositories UI

Maven2

- [NEXUS-12355⁴⁷⁴] Bug MavenModels throws an IOException when attempting to parse an empty InputStream

Migration

- [NEXUS-12081⁴⁷⁵] Bug Upgrade never completes if source repository has zero length files in it

NPM,Security

- [NEXUS-11965⁴⁷⁶] Bug npm install fails with 500 error when user has Group level privileges

Migration,NuGet

⁴⁶⁶ <https://issues.sonatype.org/browse/NEXUS-11972>

⁴⁶⁷ <https://issues.sonatype.org/browse/NEXUS-12255>

⁴⁶⁸ <https://issues.sonatype.org/browse/NEXUS-12644>

⁴⁶⁹ <https://issues.sonatype.org/browse/NEXUS-12020>

⁴⁷⁰ <https://issues.sonatype.org/browse/NEXUS-10533>

⁴⁷¹ <https://issues.sonatype.org/browse/NEXUS-12250>

⁴⁷² <https://issues.sonatype.org/browse/NEXUS-12242>

⁴⁷³ <https://issues.sonatype.org/browse/NEXUS-12334>

⁴⁷⁴ <https://issues.sonatype.org/browse/NEXUS-12355>

⁴⁷⁵ <https://issues.sonatype.org/browse/NEXUS-12081>

⁴⁷⁶ <https://issues.sonatype.org/browse/NEXUS-11965>

- [NEXUS-12483⁴⁷⁷] Bug invalid nexus 2.x NuGet repository files will cause nexus 3.x upgrade to fail with NullPointerException

NuGet

- [NEXUS-12337⁴⁷⁸] Bug NuGet queries against asset attributes can be slow due to non-optimized indexes performance
- [NEXUS-12338⁴⁷⁹] Bug query parameter names for NuGet search requests are not case-insensitive
- [NEXUS-12484⁴⁸⁰] Bug targetFramework attribute in NuGet nuspec file is rendered as Unsupported

Proxy Repository

- [NEXUS-12677⁴⁸¹] Bug proxy repository default negative cache size is too low to be effective performance
- [NEXUS-10059⁴⁸²] Bug 404 response from Nexus 2 proxying Nexus 3 due to auto-routing

Repository, Transport

- [NEXUS-12077⁴⁸³] Bug Auto-blocked proxy repository logs gigantic stack trace, doesn't say what was blocked, or why
- [NEXUS-12527⁴⁸⁴] Bug Nexus will not deliver files from the on-disk cache of a proxy repository if their metadata/artifact max age has expired and the remote is not reachable performance

Repository

- [NEXUS-10503⁴⁸⁵] Bug RepositoryManagerImpl is not thread-safe

Repository Health Check

- [NEXUS-12645⁴⁸⁶] Improvement make the Repository Health Check summary useful for discovering vulnerable components
- [NEXUS-12367⁴⁸⁷] Bug UI Danger error message when enabling RHC on a Maven Snapshot Repo.

Scheduled Tasks

⁴⁷⁷ <https://issues.sonatype.org/browse/NEXUS-12483>

⁴⁷⁸ <https://issues.sonatype.org/browse/NEXUS-12337>

⁴⁷⁹ <https://issues.sonatype.org/browse/NEXUS-12338>

⁴⁸⁰ <https://issues.sonatype.org/browse/NEXUS-12484>

⁴⁸¹ <https://issues.sonatype.org/browse/NEXUS-12677>

⁴⁸² <https://issues.sonatype.org/browse/NEXUS-10059>

⁴⁸³ <https://issues.sonatype.org/browse/NEXUS-12077>

⁴⁸⁴ <https://issues.sonatype.org/browse/NEXUS-12527>

⁴⁸⁵ <https://issues.sonatype.org/browse/NEXUS-10503>

⁴⁸⁶ <https://issues.sonatype.org/browse/NEXUS-12645>

⁴⁸⁷ <https://issues.sonatype.org/browse/NEXUS-12367>

- [NEXUS-12481⁴⁸⁸] Bug NullPointerException while rebuilding maven metadata if database operations timeout

Security

- [NEXUS-11238⁴⁸⁹] Bug Repository View - Browse permission grants too much access security
- [NEXUS-12852⁴⁹⁰] Bug certain responses may print absolute file system paths in the response

SSL

- [NEXUS-10477⁴⁹¹] Bug SSL key/trust store is not thread-safe

Support Tools

- [NEXUS-11190⁴⁹²] Bug java.nio.file.NoSuchFileException for inaccessible mounts prevents support zip generation

UI

- [NEXUS-12091⁴⁹³] Bug HTTP Proxy host name setting accepts invalid characters such as space which can prevent server start
- [NEXUS-12673⁴⁹⁴] Improvement display given roles in alphabetical order by name instead of arbitrary order compatibility
- [NEXUS-12535⁴⁹⁵] Bug Manage Privileges search dialog loses cursor focus
- [NEXUS-10774⁴⁹⁶] Bug the icon to collapse user interface feature menu can be easily confused for a back navigation button
- [NEXUS-12693⁴⁹⁷] Bug Create wildcard privilege form does not redirect to list view on success

User Token

- [NEXUS-11231⁴⁹⁸] Bug Require User Token setting not enforced security

Repository Manager 3.2.1

02/15/17

⁴⁸⁸ <https://issues.sonatype.org/browse/NEXUS-12481>

⁴⁸⁹ <https://issues.sonatype.org/browse/NEXUS-11238>

⁴⁹⁰ <https://issues.sonatype.org/browse/NEXUS-12852>

⁴⁹¹ <https://issues.sonatype.org/browse/NEXUS-10477>

⁴⁹² <https://issues.sonatype.org/browse/NEXUS-11190>

⁴⁹³ <https://issues.sonatype.org/browse/NEXUS-12091>

⁴⁹⁴ <https://issues.sonatype.org/browse/NEXUS-12673>

⁴⁹⁵ <https://issues.sonatype.org/browse/NEXUS-12535>

⁴⁹⁶ <https://issues.sonatype.org/browse/NEXUS-10774>

⁴⁹⁷ <https://issues.sonatype.org/browse/NEXUS-12693>

⁴⁹⁸ <https://issues.sonatype.org/browse/NEXUS-11231>

These notes are a compilation of improvements and significant bug fixes for Nexus Repository Manager 3.2.1.

See the [complete release notes](#)⁴⁹⁹ for all resolved issues.

New and Noteworthy

This release is heavily focused on high-priority bug fixes and support reducing upgrade impediments from Nexus Repository 2 to 3.

Known Issues Affecting Upgrades from Nexus Repository Manager 2 to 3

Along with 3.2.1, we released version of Nexus Repository 2.14.3. If you wish to upgrade from Nexus Repository Manager 2 to Nexus Repository Manager 3, you must upgrade to 2.14.3 at a minimum first. See the [2.14.3 release notes](#)⁵⁰⁰ for details.

Proxy repositories now afford options to enable circular redirects and store HTTP cookies. For example, you can enable the settings for the Oracle Maven Repository (maven.oracle.com⁵⁰¹). This achieves parity for upgrading Nexus 2.14.3 to Nexus 3.2.1. [\[NEXUS-10164\]](#)⁵⁰²

During an upgrade from Nexus Repository 3 to version 2, HTTP requests contained encoded slashes in the URL denoting the pathname of a downloaded component. These encoded slashes caused issues when version 2 ran behind a reverse proxy. This issue has been resolved. [\[NEXUS-11909\]](#)⁵⁰³

General Improvements

Blob store

- [\[NEXUS-11283\]](#)⁵⁰⁴ - Bug blob store counts inaccurate

Database

- [\[NEXUS-12336\]](#)⁵⁰⁵ - Bug repository formats lacking a groupid concept can suffer from slow db queries due to missing indexes

Docker

- [\[NEXUS-11947\]](#)⁵⁰⁶ - Bug deploying docker manifest which has unknown properties causes an entire docker package to become unusable
- [\[NEXUS-12073\]](#)⁵⁰⁷ - Bug pulling from Docker group generates error unless read access assigned directly to member

⁴⁹⁹ <https://issues.sonatype.org/jira/secure/ReleaseNote.jspa?projectId=10001&version=16869>

⁵⁰⁰ <https://sonatype.zendesk.com/knowledge/articles/115002465888/>

⁵⁰¹ <http://maven.oracle.com/>

⁵⁰² <https://issues.sonatype.org/browse/NEXUS-10164>

⁵⁰³ <https://issues.sonatype.org/browse/NEXUS-11909>

⁵⁰⁴ <https://issues.sonatype.org/browse/NEXUS-11283>

⁵⁰⁵ <https://issues.sonatype.org/browse/NEXUS-12336>

⁵⁰⁶ <https://issues.sonatype.org/browse/NEXUS-11947>

⁵⁰⁷ <https://issues.sonatype.org/browse/NEXUS-12073>

- [NEXUS-12083⁵⁰⁸] - Bug exception while executing "Purge unused docker manifests and images" task

Logging

- [NEXUS-11020⁵⁰⁹] - Bug too much DEBUG logging from com.orientechnologies.orient.core.storage.impl.local.paginated.OLocalPaginatedStorage
- [NEXUS-11518⁵¹⁰] - Bug ProxyServiceException stack trace logged at WARN when remote responds with HTTP/1.1 401

NPM

- [NEXUS-11988⁵¹¹] - Bug npm hosted repository package metadata tarball URLs incorrectly contain generated-on-request placeholder after upgrade

NuGet

- [NEXUS-12310⁵¹²] - Bug select from component query for NuGet is missing database index causing it to perform slowly

OSGI

- [NEXUS-10049⁵¹³] - Bug First time starting nexus via Docker has karaf log warn

Repository, Scalability

- [NEXUS-10759⁵¹⁴] - Bug Deleting a repository of non-trivial sizes lags and floods the log with exceptions

Scheduled Tasks

- [NEXUS-10429⁵¹⁵] - Bug Task produces WARN if "incorrect" repository is selected

Search, UI, UX

- [NEXUS-10750⁵¹⁶] - Improvement limit displayed search criteria to the formats of configured searchable repositories
- [NEXUS-11140⁵¹⁷] - Bug Elasticsearch (JDK) overflow on disks larger than 2^63

Upgrade

508 <https://issues.sonatype.org/browse/NEXUS-12083>

509 <https://issues.sonatype.org/browse/NEXUS-11020>

510 <https://issues.sonatype.org/browse/NEXUS-11518>

511 <https://issues.sonatype.org/browse/NEXUS-11988>

512 <https://issues.sonatype.org/browse/NEXUS-12310>

513 <https://issues.sonatype.org/browse/NEXUS-10049>

514 <https://issues.sonatype.org/browse/NEXUS-10759>

515 <https://issues.sonatype.org/browse/NEXUS-10429>

516 <https://issues.sonatype.org/browse/NEXUS-10750>

517 <https://issues.sonatype.org/browse/NEXUS-11140>

- [\[NEXUS-10164\]⁵¹⁸](#) - Improvement add support proxying maven.oracle.com⁵¹⁹ in nexus 3
- [\[NEXUS-11909\]⁵²⁰](#) - Bug content requests to Nexus 2 by migration agent should avoid HTTP 404 Not Found caused by URL encoding
- [\[NEXUS-12076\]⁵²¹](#) - Bug Upgrade wizard checks for hard link capability even though a different ingest method is chosen
- [\[NEXUS-12099\]⁵²²](#) - Bug gradually slowing upgrade of Nexus 2 site repositories to Nexus 3 raw repositories
- [\[NEXUS-11995\]⁵²³](#) - Bug Repository migration fails with com.fasterxml.jackson.databind.JsonMappingException: Invalid type marker byte 0xfa for expected field name (or END_OBJECT marker)

User Token

- [\[NEXUS-12230\]⁵²⁴](#) - Bug User token is deleted if external server cannot be reached

2016 Release Notes

Repository Manager 3.2.0

11/01/16

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.2.0.

See the [complete release notes](#)⁵²⁵ for all resolved issues.

New and Noteworthy

Backup & Restore

We've made two improvements to Nexus Repository for a better backup/restore experience. All content written to internal storage (the 'blob stores') is now handled in a quasi-atomic way, for better consistency of backups taken while Nexus is in use.

Secondly, there's a new scheduled task to export the content of the embedded databases. We recommend all Nexus Repository installs review their backup and restore procedures in light of these new features.

⁵¹⁸ <https://issues.sonatype.org/browse/NEXUS-10164>

⁵¹⁹ <http://maven.oracle.com/>

⁵²⁰ <https://issues.sonatype.org/browse/NEXUS-11909>

⁵²¹ <https://issues.sonatype.org/browse/NEXUS-12076>

⁵²² <https://issues.sonatype.org/browse/NEXUS-12099>

⁵²³ <https://issues.sonatype.org/browse/NEXUS-11995>

⁵²⁴ <https://issues.sonatype.org/browse/NEXUS-12230>

⁵²⁵ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=16024>

Nexus Firewall Support

For Pro installs, Nexus Repository 3.2.0 now includes integration with Nexus Firewall. This brings industry-leading security to your proxy repositories, allowing policy-driven auditing and quarantining of components.

If you're already using Nexus Firewall with Nexus Repository 2, you'll want to wait for the next release, which will have built-in support for upgrading Firewall configuration and audit/quarantine state.

Purging Orphaned Docker Images

By popular demand, we've added a task to help keep down those disk-bursting Docker repository sizes. You can now remove dangling or orphaned Docker layers on demand, or schedule this cleanup to occur automatically.

Archived log file names changes

The repository manager archived log file names now adhere to the same name pattern (ie. nexus-%d{yyyy-MM-dd}.log.gz and request-%d{yyyy-MM-dd}.log.gz) [[NEXUS-11726](#)⁵²⁶]

Scalability Improvements for Larger Numbers of Repositories

The performance team has focused on some key changes that improve the overall number of repositories that Nexus can handle.

Known Issues Upgrading to 3.2.0

Along with 3.2.0, we're releasing a bugfix version of Nexus Repository 2.14.2. If you wish to upgrade to 3.2.0, you must upgrade to 2.14.2 at a minimum first. For details, see the 2.14.2 release notes.

If you have ever scheduled a Rebuild hosted npm metadata task in 2.x, we do not recommend upgrading your npm hosted repositories to 3.x until we address [NEXUS-11988](#)⁵²⁷.

Some 3.0.x repository manager users with custom roles may unexpectedly be able to see the list of all repositories in the system in the user interface Browse area after upgrade to 3.1 and 3.2. No actual content inside of those repositories is accessible. [[NEXUS-11937](#)⁵²⁸]

If your 3.0.x install uses absolute paths (instead of the default relative paths) to locations inside the data directory, then you may need to adjust those paths using a special procedure on upgrade. [We have documented the supported procedure to move blob store locations in this scenario.](#)⁵²⁹

A reminder that custom installation property changes should be done inside of \${karaf.data}/etc/nexus.properties. DO NOT make these changes inside the example file inside your application directory at . /etc/nexus-default.properties. [[NEXUS-11733](#)⁵³⁰]

General Improvements

⁵²⁶ <https://issues.sonatype.org/browse/NEXUS-11726>

⁵²⁷ <https://issues.sonatype.org/browse/NEXUS-11988>

⁵²⁸ <https://issues.sonatype.org/browse/NEXUS-11937>

⁵²⁹ <https://sonatype.zendesk.com/hc/en-us/articles/235816228>

⁵³⁰ <https://issues.sonatype.org/browse/NEXUS-11733>

Blobstore

- [NEXUS-11950⁵³¹] Improvement implement quasi-atomic writes for all blobstore content

IQ Server

- [NEXUS-10913⁵³²] Bug "Verify connection" button on IQ server configuration page does not validate credentials

Content Selectors

- [NEXUS-11634⁵³³] Bug Search and Browse not handling leading slash properly for content selectors compatibility
- [NEXUS-11632⁵³⁴] Bug Content selector preview not handling leading slash properly

Database

- [NEXUS-11903⁵³⁵] Bug ClassCastException Cannot cast java.lang.Long to java.util.Date during deploy

Docker

- [NEXUS-11698⁵³⁶] Bug browsing hosted docker repo may trigger UnrecognizedPropertyException Unrecognized field

Docker,Scheduled Tasks

- [NEXUS-9293⁵³⁷] Improvement add a task to purge dangling docker images performance

Documentation,Upgrade

- [NEXUS-11673⁵³⁸] Bug clarify nexus-edition and nexus-features property applicability

Installer

- [NEXUS-11733⁵³⁹] Bug editing nexus-default.properties is not discouraged

Logging

- [NEXUS-11726⁵⁴⁰] Bug archived log file names should be named consistently

Maven Repository,Transport

531 <https://issues.sonatype.org/browse/NEXUS-11950>

532 <https://issues.sonatype.org/browse/NEXUS-10913>

533 <https://issues.sonatype.org/browse/NEXUS-11634>

534 <https://issues.sonatype.org/browse/NEXUS-11632>

535 <https://issues.sonatype.org/browse/NEXUS-11903>

536 <https://issues.sonatype.org/browse/NEXUS-11698>

537 <https://issues.sonatype.org/browse/NEXUS-9293>

538 <https://issues.sonatype.org/browse/NEXUS-11673>

539 <https://issues.sonatype.org/browse/NEXUS-11733>

540 <https://issues.sonatype.org/browse/NEXUS-11726>

- [NEXUS-10234⁵⁴¹] Bug Maven httpclient may receive SocketException Broken pipe instead of expected status code on deploy

Maven Repository,Scheduled Tasks

- [NEXUS-11614⁵⁴²] Bug poor performance from snapshot removal tasks rebuilding maven metadata performance

Migration

- [NEXUS-11925⁵⁴³] Bug Nexus 2 to Nexus 3 upgrade may fail with NullPointerException Cannot invoke method extract while processing RepositoryChangelogResource
- [NEXUS-11923⁵⁴⁴] Improvement perform version compatibility check when upgrading from Nexus 2.x to Nexus 3.x
- [NEXUS-11732⁵⁴⁵] Bug prevent ClassCastException when handling StorageLinkItem during upgrade to Nexus 3
- [NEXUS-11921⁵⁴⁶] Improvement explicitly prevent quarantine enabled Nexus 2 repositories from upgrade into Nexus 3
- [NEXUS-11301⁵⁴⁷] Bug upgrade to Nexus 3 fails if Nexus 2 has no anonymous user defined

Search

- [NEXUS-10638⁵⁴⁸] Improvement add a search criteria for repository name

UI

- [NEXUS-9500⁵⁴⁹] Improvement make capability validation message for URL type fields consistent and less confusing
- [NEXUS-9302⁵⁵⁰] Bug Administration cog icon may not be initially visible after successful sign in

Repository Manager 3.1.0

09/12/16

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.1.0.

541 <https://issues.sonatype.org/browse/NEXUS-10234>

542 <https://issues.sonatype.org/browse/NEXUS-11614>

543 <https://issues.sonatype.org/browse/NEXUS-11925>

544 <https://issues.sonatype.org/browse/NEXUS-11923>

545 <https://issues.sonatype.org/browse/NEXUS-11732>

546 <https://issues.sonatype.org/browse/NEXUS-11921>

547 <https://issues.sonatype.org/browse/NEXUS-11301>

548 <https://issues.sonatype.org/browse/NEXUS-10638>

549 <https://issues.sonatype.org/browse/NEXUS-9500>

550 <https://issues.sonatype.org/browse/NEXUS-9302>

See the [complete release notes](#)⁵⁵¹ for all resolved issues.

New and Noteworthy

Building on the solid OSS foundation, Nexus Repository Manager 3.1.0 is the first PRO release of the 3.x platform and the first 3.x release to support upgrade from 2.x.

Upgrade from Nexus 2.14.1 is supported!

Nexus Repository Manager 3.1.0 is the first 3.x series release that supports upgrade from *Nexus 2.14.1 and greater*. Please review our [upgrade guide](#)(see page 146) for more information on the exact steps required.

Changes To Default Installation and Work Directory Layout

Version 3.1 changes the default locations of core installation files relative to your specific work and data. It now more closely mimics the familiar pattern that Nexus 2.x followed. For users upgrading from 3.0.1/3.0.2, we strongly recommend adopting these defaults by following [our special upgrade instructions](#)⁵⁵².

Common Server Customizations More Easily Persisted Across Upgrades

Now you can store your server configuration files changes directly in the data directory, making backups and upgrades straight forward. Upgrades will involve fewer manual changes while easily picking up sensible default values for any new features.

Upgrading from 3.0.x? Follow these Instructions

We have [special instructions for users upgrading their 3.0 installs to 3.1](#)⁵⁵³.

Nexus Professional Features Support

Existing Pro customers can install their licenses into this version and begin to take advantage of advanced enterprise features.

UI Based Installer Distribution is Suspended

For the near future we have decided to remove the UI based installers. If you have previously used these installers, please [review our article](#)⁵⁵⁴ for more information.

Security Issues Resolved

As part of our continuous auditing, a number of potential UI exploits and vulnerable third-party dependencies have been fixed in this release to make a more secure product.

General Improvements

Blobstore

- [\[NEXUS-10446\]](#)⁵⁵⁵ Story blob store names should be case insensitive

⁵⁵¹ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=15624>

⁵⁵² <https://sonatype.zendesk.com/hc/en-us/articles/231723267>

⁵⁵³ <https://sonatype.zendesk.com/hc/en-us/articles/231723267>

⁵⁵⁴ <https://sonatype.zendesk.com/hc/en-us/articles/229935227>

⁵⁵⁵ <https://issues.sonatype.org/browse/NEXUS-10446>

- [NEXUS-9873⁵⁵⁶] Story If there is only one blobstore then just select it in the new repository screen

Bootstrap

- [NEXUS-10418⁵⁵⁷] Improvement automatically clear the karaf bundle cache on startup
- [NEXUS-10454⁵⁵⁸] Story Move the default location of the data directory in the tar/zip installs out of the install directory

Configuration

- [NEXUS-10829⁵⁵⁹] Story Store commonly customized configuration in the data folder

Database

- [NEXUS-10154⁵⁶⁰] Bug OrientDb allocates massive disk cache on large systems performance
- [NEXUS-10518⁵⁶¹] Story Upgrade to OrientDB 2.2.x

Docker

- [NEXUS-10630⁵⁶²] Bug Unable to push image whenever it shares layers with another image already pushed by another docker client
- [NEXUS-10147⁵⁶³] Bug add ability to dump invalid JSON payloads submitted by docker on parse errors

Logging

- [NEXUS-10883⁵⁶⁴] Bug DEBUG level logging should print HTTP response code Nexus is sending
- [NEXUS-10242⁵⁶⁵] Story Capture stdout and stderr to logfile

NPM

- [NEXUS-10817⁵⁶⁶] Bug publishing npm packages with wrongly encoded ISO-8859-1 JSON fails with 400

NuGet

- [NEXUS-8941⁵⁶⁷] Bug Nuget packageId case sensitivity

556 <https://issues.sonatype.org/browse/NEXUS-9873>

557 <https://issues.sonatype.org/browse/NEXUS-10418>

558 <https://issues.sonatype.org/browse/NEXUS-10454>

559 <https://issues.sonatype.org/browse/NEXUS-10829>

560 <https://issues.sonatype.org/browse/NEXUS-10154>

561 <https://issues.sonatype.org/browse/NEXUS-10518>

562 <https://issues.sonatype.org/browse/NEXUS-10630>

563 <https://issues.sonatype.org/browse/NEXUS-10147>

564 <https://issues.sonatype.org/browse/NEXUS-10883>

565 <https://issues.sonatype.org/browse/NEXUS-10242>

566 <https://issues.sonatype.org/browse/NEXUS-10817>

- [\[NEXUS-11141\]⁵⁶⁸](https://issues.sonatype.org/browse/NEXUS-11141) Improvement support for nuget repository "package-versions" endpoint
- [\[NEXUS-10808\]⁵⁶⁹](https://issues.sonatype.org/browse/NEXUS-10808) Bug eager caching of nuget versions contributes to slow query performance performance

Repository

- [\[NEXUS-10473\]⁵⁷⁰](https://issues.sonatype.org/browse/NEXUS-10473) Bug tmp files may be kept open after deletion performance

Repository,UI

- [\[NEXUS-10019\]⁵⁷¹](https://issues.sonatype.org/browse/NEXUS-10019) Bug Deletion of asset from single asset component causes failure in UI
- [\[NEXUS-9844\]⁵⁷²](https://issues.sonatype.org/browse/NEXUS-9844) Bug Wrong max component age pre-set for release Maven proxy

Scheduled Tasks,UI

- [\[NEXUS-9523\]⁵⁷³](https://issues.sonatype.org/browse/NEXUS-9523) Bug Starting to leave the browser with a dirty page but staying can break the UI

Search

- [\[NEXUS-11254\]⁵⁷⁴](https://issues.sonatype.org/browse/NEXUS-11254) Bug task "Rebuild repository index" may fail with with OutOfMemoryError

Security

- [\[NEXUS-10882\]⁵⁷⁵](https://issues.sonatype.org/browse/NEXUS-10882) Improvement upgrade Apache Shiro dependency to 1.3.1 to pick up recent security fixes

UI

- [\[NEXUS-10917\]⁵⁷⁶](https://issues.sonatype.org/browse/NEXUS-10917) Bug Repository Combobox Filtering not working

UX

- [\[NEXUS-10922\]⁵⁷⁷](https://issues.sonatype.org/browse/NEXUS-10922) Story Mask the UI on repository/blob store deletion

Webhooks

- [\[NEXUS-10823\]⁵⁷⁸](https://issues.sonatype.org/browse/NEXUS-10823) Improvement component / asset webhook events for common operations
-

⁵⁶⁸ <https://issues.sonatype.org/browse/NEXUS-11141>

⁵⁶⁹ <https://issues.sonatype.org/browse/NEXUS-10808>

⁵⁷⁰ <https://issues.sonatype.org/browse/NEXUS-10473>

⁵⁷¹ <https://issues.sonatype.org/browse/NEXUS-10019>

⁵⁷² <https://issues.sonatype.org/browse/NEXUS-9844>

⁵⁷³ <https://issues.sonatype.org/browse/NEXUS-9523>

⁵⁷⁴ <https://issues.sonatype.org/browse/NEXUS-11254>

⁵⁷⁵ <https://issues.sonatype.org/browse/NEXUS-10882>

⁵⁷⁶ <https://issues.sonatype.org/browse/NEXUS-10917>

⁵⁷⁷ <https://issues.sonatype.org/browse/NEXUS-10922>

⁵⁷⁸ <https://issues.sonatype.org/browse/NEXUS-10823>

Repository Manager 3.0.2

07/06/16

These notes are a compilation of new features and significant bug fixes for Nexus Repository Manager 3.0.2.

See the [complete release notes](#)⁵⁷⁹ for all resolved issues.

New and Noteworthy

Python PyPi and RubyGems Repository Format Support Added

Python PyPi and RubyGems developers can now enjoy the use of a world class repository manager for free in repository manager 3.0.2. Visit our documentation sections for more details.

Did we mention these repository formats are available for free in Nexus Repository Manager OSS 3.0.2?

Docker Related Fixes

Some less frequently encountered but otherwise important Docker related bugs are fixed in this release. Upgrade to make sure you do not hit these.

Potential HTTP Connection Leak Fixed

All proxy repositories have the potential to leak HTTP connections - this is mostly noticeable under sustained proxy repository load and frequent outbound requests. The symptoms may include having the repository manager non-responsive to inbound requests, thread backup and ParOldGen filling up and not being reclaimed.

General Improvements

Bower

- [NEXUS-10564]⁵⁸⁰ **Bug** Bower proxy repository timeout waiting for connection from pool performance

Configuration

- [NEXUS-10021]⁵⁸¹ **Bug** Authentication-less Email Server config does not work

Docker

- [NEXUS-10294]⁵⁸² **Bug** Docker client <= 1.9.1 cannot pull images which are push by a docker client >= 1.10.0
- [NEXUS-10421]⁵⁸³ **Bug** Docker pull through proxy repository fails if remote is not available

⁵⁷⁹ <https://issues.sonatype.org/secure/ReleaseNote.jspa?projectId=10001&version=16520>

⁵⁸⁰ <https://issues.sonatype.org/browse/NEXUS-10564>

⁵⁸¹ <https://issues.sonatype.org/browse/NEXUS-10021>

⁵⁸² <https://issues.sonatype.org/browse/NEXUS-10294>

- [NEXUS-10481⁵⁸⁴] **Bug** outbound docker proxy requests always insert /library/ and commonly return 404 at the remote

NPM

- [NEXUS-10423⁵⁸⁵] **Bug** NPM proxy repository Timeout waiting for connection from pool performance

PyPi

- [NEXUS-6037⁵⁸⁶] **Improvement** Python PyPi repository format support

RubyGems

- [NEXUS-10827⁵⁸⁷] **Improvement** RubyGems repository format support

Security

- [NEXUS-7765⁵⁸⁸] **Bug** RUT Auth Realm does not authenticate in Nexus 3

Support Tools

- [NEXUS-10042⁵⁸⁹] **Bug** IOException Pipe not connected prevents generating support zip on Windows

Transport

- [NEXUS-10390⁵⁹⁰] **Bug** Nexus 3 Basic realm name does not equal Nexus 2 realm name
-

Repository Manager 3.0.1

07/06/16

New and Noteworthy

This is a bug fix release only.

If you intend to upgrade to this release, please carefully follow our instructions on [How to Upgrade Nexus Repository Manager 3](#)⁵⁹¹.

General Improvements

Database,NuGet

⁵⁸⁴ <https://issues.sonatype.org/browse/NEXUS-10481>

⁵⁸⁵ <https://issues.sonatype.org/browse/NEXUS-10423>

⁵⁸⁶ <https://issues.sonatype.org/browse/NEXUS-6037>

⁵⁸⁷ <https://issues.sonatype.org/browse/NEXUS-10827>

⁵⁸⁸ <https://issues.sonatype.org/browse/NEXUS-7765>

⁵⁸⁹ <https://issues.sonatype.org/browse/NEXUS-10042>

⁵⁹⁰ <https://issues.sonatype.org/browse/NEXUS-10390>

⁵⁹¹ <https://sonatype.zendesk.com/hc/articles/217967608>

- [NEXUS-10278⁵⁹²] Bug pushing NuGet packages larger than 2GB fails

Docker

- [NEXUS-10166⁵⁹³] Bug RedHat docker 1.8.2 push HTTP PUT uploads tar content instead of gzip and triggers JsonParseException 400
- [NEXUS-10148⁵⁹⁴] Bug docker V2 pull requests may return V1 manifests unexpectedly resulting in missing signature key error

Docker,Proxy Repository

- [NEXUS-10115⁵⁹⁵] Bug Docker proxy repository ConnectionPoolTimeoutException Timeout waiting for connection from pool

LDAP,Security

- [NEXUS-10256⁵⁹⁶] Bug extremely poor performance in user role/privilege resolution for LDAP users

NPM

- [NEXUS-10069⁵⁹⁷] Bug npm install against a group repository with nested group member fails with 500 status

NuGet

- [NEXUS-9502⁵⁹⁸] Bug NuGet proxy repository responds with 502 status instead of 404 for missing content

Proxy Repository,UI

- [NEXUS-9993⁵⁹⁹] Bug proxy repository HTTP request settings form fields cannot be saved

Repository

- [NEXUS-10044⁶⁰⁰] Bug StackOverflowError when a group repository has itself as a member

Repository Health Check,Scheduled Tasks

592 <https://issues.sonatype.org/browse/NEXUS-10278>

593 <https://issues.sonatype.org/browse/NEXUS-10166>

594 <https://issues.sonatype.org/browse/NEXUS-10148>

595 <https://issues.sonatype.org/browse/NEXUS-10115>

596 <https://issues.sonatype.org/browse/NEXUS-10256>

597 <https://issues.sonatype.org/browse/NEXUS-10069>

598 <https://issues.sonatype.org/browse/NEXUS-9502>

599 <https://issues.sonatype.org/browse/NEXUS-9993>

600 <https://issues.sonatype.org/browse/NEXUS-10044>

- [NEXUS-9639⁶⁰¹] Bug startup race condition ClassNotFoundException `javax.ws.rs-api` RHC can prevent startup

Scheduled Tasks, UI

- [NEXUS-9995⁶⁰³] Bug scheduled task form values cannot be blanked out when a blank value is valid

Support Tools

- [NEXUS-10102⁶⁰⁴] Bug generating support zip with configuration option requires external internet access

UI

- [NEXUS-9991⁶⁰⁵] Bug ui spinbutton form controls should not react to mouse scroll wheel movement
-

Repository Manager 3.0.0

03/10/16

We are pleased to announce our newest version of Nexus Repository Manager. Our team is dedicated to delivering enterprise-quality software for you to manage all your component binaries in one tool. That's why we have gone beyond our previous versions to give you improvements to user interface, system performance, and component format support. With all these new features in place, we are delighted to introduce Nexus Repository Manager 3.0!

Before You Begin

1. Currently, users cannot migrate an existing Nexus Repository Manager 2 install to version 3. The first version of a migration tool will be available in 3.1. Read more about our current [features⁶⁰⁶](#) and what's coming soon. You can see a demo video and read about migration on [the Nexus community site⁶⁰⁷](#).
2. The new repository manager requires Java Runtime Environment 8, at minimum. Upgrade if needed.
3. Experience with installing and running the Nexus Repository Manager 2 and repository concepts can be helpful, but not required.

What's New and Noteworthy for 3.0.0?

Here is list of all noteworthy changes to Nexus Repository Manager 3.0.0:

- Platform-specific installer application

601 <https://issues.sonatype.org/browse/NEXUS-9639>

602 <http://javax.ws/>

603 <https://issues.sonatype.org/browse/NEXUS-9995>

604 <https://issues.sonatype.org/browse/NEXUS-10102>

605 <https://issues.sonatype.org/browse/NEXUS-9991>

606 <http://www.sonatype.org/nexus/2016/04/06/spring-into-the-future-nexus-repository-manager-3-0-release/>

607 <http://www.sonatype.org/nexus/2016/05/11/migrating-to-nexus-repository-3-easy-peasy/>

- Re-designed user interface and navigation
- 'Remember Me' feature to prevent re-logging into the UI over and over
- New OSGI console, based on the Apache Karaf platform
- View component and asset details
- Ability to schedule tasks
- Ability to bookmark search results
- Repository support for [Bower\(see page 371\)](#), [Docker\(see page 383\)](#), [Maven2\(see page 411\)](#), [NuGet\(see page 428\)](#), and [npm\(see page 422\)](#)
- Improved security (Realms) for NuGet and npm
- Enhanced security for Privilege creation, allowing more flexible permissions for accounts
- Ability to search across all repository formats and components
- [Repository Health Check](#)⁶⁰⁸ for Maven, npm, NuGet proxy repositories
- Ability to specify SSL and TLS options for email configuration
- Large performance improvements to our Blobstore
- New default URL pointing to `http://localhost:8081/`, previously `http://localhost:8081/nexus` in v2
- [Java API\(see page 464\)](#) for provisioning and configuration

Should I Use Version 3 or 2?

We are advising that you use Nexus Repository 3.0 if you are planning to use npm, Bower, NuGet or Docker. If you are using Maven, we suggest that you continue to use Nexus Repository 2.x as we are still building features for 3.0 that many Maven folks will find handy. If you want to run both versions in parallel, that is perfectly fine too.

How Do I Upgrade?

Nexus Repository Manager 3 Milestone 7 is the only milestone release that can be updated to 3.0.0 Final. Follow these instructions to perform the [upgrade](#)⁶⁰⁹.

Tell Us What You Think

We hope you will take this opportunity to discuss your experience with Nexus Repository Manager 3.0. To do this, just send us an email at nexus-feedback@sonatype.com⁶¹⁰ or contact us via the [mailing list](#) or [chat](#)⁶¹¹.

Milestone Releases

⁶⁰⁸ <https://sonatype.wistia.com/medias/77jh7h47av>

⁶⁰⁹ <https://sonatype.zendesk.com/hc/en-us/articles/217967608-How-to-Upgrade-Nexus-3-Milestone-m7-to-3-0-0-Final>

⁶¹⁰ <mailto:nexus-feedback@sonatype.com>

⁶¹¹ <http://www.sonatype.org/nexus/lists-live-chat-and-other-resources/>

⚠ You are viewing out-of-date release notes. Please [download\(see page 22\)](#) and use the latest version of Nexus Repository Manager 3.

Milestone 7 Release

Jan 2016: We have all been working hard towards this release and this is it! From now on, you can upgrade your Nexus Repository Manager 3 deployment! This means that you can start using the shiny new Milestone 7 release in production. It is ready to roll out and includes features that go beyond what version 2 has to offer Docker and npm users.

Of course we still have a long list of new features in the pipeline for you with the next releases as well. For now though, this release includes:

- Improved support for users of Maven repositories and tools like sbt and Apache Ivy.
- Repository health check with expanded coverage including NuGet and npm repositories.
- Support for upgrading from this version going forward.

Curious and can't wait? Download a bundle and do a fresh install of the release right now. And if you want to be a little more cautious, read about what's new and what to be careful about .

Regardless of the path you choose, we hope you will take this opportunity to discuss your experience with this latest release. Drop us an email at nexus-feedback@sonatype.com⁶¹² or contact us [via the mailing list](#) or [chat](#)⁶¹³.

Have fun!

The Nexus Repository Manager Team

Before you begin

Some important things to keep in mind:

1. Upgrades from previous Milestones are NOT supported. Perform a fresh install instead.
2. This Milestone WILL BE upgrade-able to the future 3.0 release, so go ahead and get real with this one.
3. Experience with installing and running the Nexus Repository Manager 2 and repository concepts can be helpful, but is not required.
4. We consider the Docker, NPM (v2) and NuGet implementations to be stable and ready for production use to complement Nexus v2 which still has more Maven specific functionality. Though this code is considered stable, it is still an early version. We recommend vigilance in following the guidelines regarding backup and restore such that there is a recover path in the event that it is needed.

⁶¹² <mailto:nexus-feedback@sonatype.com>

⁶¹³ <http://www.sonatype.org/nexus/lists-live-chat-and-other-resources/>

5. Installation and configuration processes have changed from the milestone 6 release, refer to the documentation for details

What's New?

The significant new features of this milestone release are:

- Improvements to our Maven 2 format support
- Repository Health Check for Maven, npm, NuGet proxy repos
- Upgrade-ready, Blobstores, and SSL

Maven Repository Improvements

We've added quite a few items for Maven repositories in this release including:

- A few more Scheduled Tasks to help you keep the Nexus Repository Manager up and humming
- Relaxing the repository format such that you can publish from SBT, Ivy or other build tools/formats that don't match Maven verbatim
- Support for Maven archetype catalog
- Support for generating and downloading Maven .index files

If you are a Maven repository user, give us your [feedback](#)⁶¹⁴.

Expanding the Repository Health Check

You can now run [Repository Health Check](#)⁶¹⁵ against your Maven, npm and NuGet proxy repositories in this release and beyond. While this is functionality that existed in version 2, we are interested in your experience running it against proxy repositories in version 3.

Did you know that we run Repository Health Checks on over 50,000 repositories each day. Why not give it a try?

Blobstores and More

Some other cool things we added to this release:

- Upgrades will now be supported from this Milestone to future versions of Nexus Repository Manager 3.
- Ability to specify SSL and TLS options for email configuration
- Improvements to store Blobstore paths as relative instead of absolute
- Large performance improvements to our Blobstore

⁶¹⁴ mailto:nexus-feedback@sonatype.com

⁶¹⁵ <http://blog.sonatype.com/how-to-use-the-new-repository-health-check-2.0>

Milestone 6 Release

Nov 2015: We have been listening to you! Our last milestone 5 release included Docker support and the feedback we got from users was very positive. One popular request was to add support for the Docker Registry API V2. So we went ahead and did that and a bunch of other Docker related improvements.

But of course we did not stop there. We implemented npm support for proxying, hosting and grouping repositories for users of the npm package manager popular for Node.js development and other JavaScript related work. Now you are going to say we already had that in Nexus 2. True, but this time we also support searching and browsing npm repositories, npm login, npm deprecate and npm scopes and more. You are gonna love it.

And to make it easier for you to get started, we refactored the archive bundles we ship to include the JRE needed and added a convenient installer application for your operating system of choice.

Curious and can't wait? Download a bundle and do a fresh install of the release right now. And if you want to be a little more cautious, read about what's new and what to be careful about.

Regardless of the path you choose, we hope you will take this opportunity to discuss your experience with this latest release. Drop us an email at nexus-feedback@sonatype.com⁶¹⁶ or contact us [via the mailing list](#) or [chat](#)⁶¹⁷.

Have fun!

The Nexus Repository Manager Team

Before you begin

Some important things to keep in mind:

1. Upgrades between milestone builds are NOT supported. Perform a fresh install instead.
2. Experience with installing and running the Nexus Repository Manager 2 and repository concepts can be helpful, but is not required.
3. We consider the Docker implementation to be stable and ready for production trials. Though this code is considered stable, it is still an early version. We recommend vigilantly following the guidelines regarding backup and restore such that there is a recover path in the event that it is needed.
4. Installation and configuration processes have changed from the milestone 5 release, refer to the documentation for details.

Milestone Backup and Restore

Comprehensive backup and restore procedures are being worked on. Milestone releases can follow this basic procedure for now.

Backup

⁶¹⁶ mailto:nexus-feedback@sonatype.com

⁶¹⁷ <http://www.sonatype.org/nexus/lists-live-chat-and-other-resources/>

While Nexus is stopped, copy the entire install directory to another location.

Restore

Start Nexus from a backed up copy.

What's New?

If this is your first time trying a Nexus Repository Manager 3 milestone release, check out our notes for previous releases. These are all available here in our [Release Notes\(see page 37\)](#) pages. It will give you a good background for any of the previous changes, especially those related to the user interface as well as installation procedures.

The significant new features of this milestone release are:

Installer Application and Java 8

The new download archives are available for OSX, Windows and Linux and include the required Java 8 runtime. A new, user-friendly installer application allows anybody to follow a few steps using a simple user interface and install the repository manager in minutes. If you are installing and starting from the archive, you have to keep in mind that the launcher script usage changed a bit as well E.g. use `./bin/nexus run` now.

Docker Registry API V2 Support

Implementing support for the Docker Registry API V2 required us to read Docker source code, reverse engineer behaviour of DockerHub and other registries and generally get into the nitty-gritty details of it all. And we wrapped it all up in an easy to use configuration for you so you can use V2 and fall back to V1 seamlessly when required. At least for searching in DockerHub via command-line you still want to have V1 enabled, since DockerHub itself still uses only V1 for that. Anyway - enough details. Give it a whirl and [check out the documentation\(see page 383\)](#), when necessary.

Node Packaged Modules and npm Registries

[Support for npm packages managed in Nexus Repository Manager-controlled repositories\(see page 422\)](#) is better than ever before. Proxy `npmjs.org` to avoid repeated downloads and improve your build performance, create npm repository hosted on the repository manager as a private registry and share your packages with internally to your organization with ease. Use npm login to easily authenticate to the repository for publishing. Or use npm deprecate to let your users know about a new release of your shared package. Or simply search or browse in the repository manager to see what is available. Once you got used to managing your npm packages with the repository manager, you won't believe you used npm without it in the past.

Milestone 5 Release

21 Sept. 2015: With giant steps we are moving forward towards the release of Nexus 3.0. The new Milestone 5 release adds support for a completely new repository format - Docker. Just with all other formats you can proxy and host repositories. For Docker users in the DevOps realm this is a significant release since it allows you to run on-premises, private Docker registries with Nexus with an open source solution. No other product out there offers this.

And it was all built on top of the format-agnostic storage system introduced in the last milestone release. We have further improved it by adding features such as browsing repositories and accessing component details in the Nexus user interface.

Beyond that we improved and introduced numerous smaller features related to the user interface, tasks and other functionality of Nexus.

Curious and can't wait? Download a Nexus OSS bundle and do a fresh install of the release right now. And if you want to be a little more cautious, read about what's new and what to be careful about.

Regardless of the path you choose, we hope you will take this opportunity to discuss your experience with this latest release. Drop us an email at nexus-feedback@sonatype.com⁶¹⁸ or contact us [via the mailing list or chat](#)⁶¹⁹.

Have fun!

The Sonatype Nexus Team

Before you begin

Some important things to keep in mind:

1. Upgrades between milestone builds are NOT supported. Perform a fresh install instead.
2. Experience with installing and running Nexus 2 and repository concepts can be helpful, but is not required.
3. Do NOT use Nexus 3 milestone builds on critical, production servers.

What's New?

If this is your first time trying a Nexus 3 milestone release, check out our notes for [previous releases\(see page 37\)](#). These are all available here in our [Release Notes\(see page 37\)](#) pages. It will give you a good background for any of the previous changes, especially those related to the user interface as well as installation procedures.

The significant new features of this milestone release are:

Docker

Docker container usage is revolutionizing development and operations workflows everywhere. The benefits are tremendous. This milestone release of Nexus 3 adds full support for Docker registries. You can configure proxy repositories of Docker Hub and any other repository. This allows you to avoid repeated downloads of Docker images and layers, by caching them in Nexus running in your network. Creating a hosted repository enables you to have completely private Docker repositories within your organization enabling the exchange of proprietary images between development teams as well as for QA and production usage. All those repositories can be wrapped in a repository group, so that your Docker users still have to deal with only one URL and gain access to a much larger set of images.

⁶¹⁸ mailto:nexus-feedback@sonatype.com

⁶¹⁹ <http://www.sonatype.org/nexus/lists-live-chat-and-other-resources/>

Beyond these features we have simplified the required SSL setup for Nexus and added support for Docker specific search. And of course Repository Browsing and Component and Asset Detail views work for Docker as well...

Repository Browsing

You want to know what's in all those different repositories? What got proxied from nuget.org? Or the Central Repository? What release got deployed to your hosted repository? What Docker images have been pulled from Docker Hub? All these questions and many others can now be answered with the new repository browsing user interface in Nexus. You can access all the components and their assets per repository.

Component and Asset Details

You can browse your repository or find components via the search functionality, but now what. With the new component and asset details view you can see all the details about each component including the repository format agnostic characteristics like name and version, but also all the different attributes used in the different formats and associated to the components and assets.

Known Issues

We are currently working to resolve an issue that specifically impacts the ability to browse repositories. When first navigating to a repository, the list of components or assets may appear blank until you either scroll, refresh, or come back to the page.

For this milestone, we have added Docker Registry v1 capabilities so that we could get early feedback on how Docker fits within Nexus. Docker v2 specific features are currently unavailable though are being worked as part of the next milestone and will likely be complete as part of the GA release. A full list of v2 features is available [here](#). We plan on getting feedback from the community along the way to help fine tune how we support you all into the future.

Milestone 4 Release

It's been a little while, since we released our last milestone! We promise the wait was worth it.

In this milestone 4 release we have gone behind the scenes and changed a lot of the backend systems used to store components, metadata and configuration. We built upon years of experience and designed a system that scales well into the future and the demands to support numerous, different repository formats with a large variety of components and metadata.

And of course, we have continued with our efforts to create a slick, modern web application that meets your demands in terms of performance and usability.

Curious and can't wait? Download a Nexus OSS bundle and do a fresh install of the release right now. And if you want to be a little more cautious, read about what's new and what to be careful about.

Regardless of the path you choose, we hope you will take this opportunity to discuss your experience with this latest release. Drop us an email at nexus-feedback@sonatype.com⁶²⁰ or contact us [via the mailing list or chat](#)⁶²¹.

And if you want to know more, check out [our story on TheNexus](#) and have a look at the quick video demo [there to see it in action](#)⁶²².

Have fun!

The Sonatype Nexus Team

Looking for a few good Nexans (Nexus users)

The great state of Nexus is looking for participants to help us validate the new functionality in Milestone 4. Each session will run about 45 minutes. You'll get to try a variety of tasks using the release and tell us what works, and what doesn't. We'll use this feedback to eliminate any glaring holes and polish the rough edges. You get to say you had a hand in improving a product you use every day.

If interested, send us an email at nexus-feedback@sonatype.com⁶²³. We'll schedule your session to occur within the next couple of weeks. If you choose to participate, please don't look at the release beforehand. We want your initial reaction!

We look forward to hearing from you,

The Sonatype User Experience Team

Before you begin

Some important things to keep in mind:

1. Upgrades between milestone builds are NOT supported. Perform a fresh install instead.
2. Experience with installing and running Nexus 2 and repository concepts can be helpful, but is not required.
3. Do NOT use Nexus 3 milestone builds on critical, production servers.

What's New?

This milestone represents significant changes "under the hood", mainly in the form of an updated component and metadata storage framework. Beyond that far reaching change and the affected user interface sections, we put further efforts into the user experience.

⁶²⁰ mailto:nexus-feedback@sonatype.com

⁶²¹ <http://www.sonatype.org/nexus/lists-live-chat-and-other-resources/>

⁶²² <http://www.sonatype.org/nexus/2015/06/12/behind-the-scenes-of-the-shiny-new-nexus-3-milestone-4-release/>

⁶²³ mailto:nexus-feedback@sonatype.com

⚠ If this is your first time trying a Nexus 3 milestone release, check out our notes for previous releases. These are all available here in our [Release Notes\(see page 37\)](#) pages. This will give you a good background for any of the previous changes, especially those related to the user interface as well as installation procedures.

Storage Backend

While physicists around the world are trying to create a [Grand Unified Theory](#)⁶²⁴, we have pulled ahead and implemented a grand unified component storage system.

Figure 1. It's hard to show a backend... so here's some robutt.

This new backend is independent of the repository format and flexible enough to support any component and metadata format. For starters we have implemented support for the repository formats used by NuGet as well as Maven. And we also added a very simple raw repository format that can e.g. be used to host Maven-generated sites. The changes of the underlying semantics in turn allowed us to improve the repository management user interface as well as search.

Search

A major benefit of the new storage backend is its flexibility. It allows search to work across all repository formats and components and include all the metadata. We have added a number of search criteria and built a custom search interface as well as pre-defined searches for specific use cases. Screenshots and more information can be found in the [documentation\(see page 0\)](#).

Repository Administration

The updated repository administration user interface sits on top of the storage backend and allows you to manage your repositories in maven2, nuget and raw formats. All formats support proxying remote repositories, direct storage of components in Nexus hosted repositories as well as grouping repositories. The user interface clearly separates the different aspects of repository configuration and makes your repository management easy and [the documentation tells you more about all the details\(see page 0\)](#).

Tab Consolidation

A new drill-down pattern in the Nexus 3 user interface allows you to focus on the current content and feature and reduces the visual complexity and number of tabs.

Milestone 3 Release

We're still hard at work on Nexus 3, but thought we would pause and give you a glimpse of the current state of development. As with previous milestones, Milestone 3 builds on existing functionality, adding significant

⁶²⁴ https://en.wikipedia.org/wiki/Grand_Unified_Theory

improvements to the visual design and overall experience. The emphasis has been on usability and visual polish, setting the foundation for future UI improvements.

Once you've read through our disclaimers, feel free to download and install the release. An overview of what's new is below, for the curious.

Most importantly, please let us know what you think of the release. Are you experiencing any problems? Are there aspects of the release that you love or hate?

Let us know. Drop us an email at nexus-feedback@sonatype.com⁶²⁵.

A word of CAUTION! before you begin

Some important things to keep in mind:

1. You should be experienced installing and administering Nexus already. If not, we recommend starting with the latest version of Nexus 2.xx.
2. It probably goes without saying, but do NOT upgrade a production server with any milestone builds.
3. Upgrades between milestone builds may NOT be fully supported.
4. Fully updated documentation is not currently available, however we will be providing some items to help you get started. (See below)
5. Have fun, tell us what you like, what you don't, and if you are so inclined, try to break things.

What's New?

⚠ If this is your first time trying a Nexus 3 milestone release, check out our notes for [Milestone 1](#)(see [page 133](#)). This will give you a good background for many of the changes you can expect, especially those related to the user interface.

As mentioned, this release is mostly about improvements to the user interface. None of these changes are necessarily final, but they reflect the general direction we're taking the UI. As always, we want your feedback at nexus-feedback@sonatype.com⁶²⁶.

Detail panels have more breathing room

Previously, detail panels cut the available space in half. For busy features, this resulted in a cramped experience. To fix this, we implemented what we call "Drilldown", a pattern which uses sliding full-page panels accompanied by a breadcrumb for navigation.

Search results can be bookmarked

In MS2, there was no way to search for a component, view details for a specific version, and copy/paste a URL for that. This has been fixed in MS3.

Fewer modals, more full-page screens

⁶²⁵ <mailto:nexus-feedback@sonatype.com>

⁶²⁶ <mailto:nexus-feedback@sonatype.com>

Where it made sense, we opted to use the drilldown pattern instead of modals. One prominent example is that all create modals are now drilldowns, which gives the associated lists and forms more space on the page.

Numerous style tweaks

Too many small improvements to list, but some highlights:

- Page organized with typography, not background color.
 - Header mode buttons are full height and have higher contrast.
 - Colors normalized to a well-defined palette
 - Main font changed to Proxima Nova
 - Login modal simplified
 - Tabs replaced with pills
 - Inline frames replaced with borderless containers and large titles
 - Help text across the experience was simplified
 - Focus states made visible for accessibility purposes
-

Milestone 2 Release

The Nexus Team is proud to announce our latest milestone release, Milestone 2, which now includes all the features of Nexus Pro, and all of the updates we made in Nexus 3 Milestone 1.

Thus far, the focus has been the new user interface (and still is). However, there are numerous refinements to the internals of Nexus not readily apparent in the new look and feel.

If you have already tried out the Milestone 1 build, there won't be a lot of changes here. That is, beyond the Nexus Pro features.

Because of this, we're going to go over very similar content covered in the [Milestone 1 announcement\(see page 133\)](#). If you would simply like to skip ahead and download the latest Milestone release, go right ahead, we won't blame you.

But... be sure to check out our word of CAUTION below, as well as some of the other tips for successfully trying this sample of features for the upcoming version of Nexus 3.

As always, thank you, and [let us know what you think](#)⁶²⁷.

A word of CAUTION! before you begin

Some important things to keep in mind:

1. You should be experienced installing and administering Nexus already. If not, we recommend starting with the latest version of Nexus 2.xx.
2. It probably goes without saying, but do NOT upgrade a production server with any milestone builds.
3. Upgrades between milestone builds may NOT be fully supported.

⁶²⁷ mailto:nexus-feedback@sonatype.com

4. Fully updated documentation is not currently available, however we will be providing some items to help you get started. (See below)
5. Have fun, tell us what you like, what you don't, and if you are so inclined, try to break things.

What if I experience problems, or want to offer feedback?

Great, we'd love to hear from you. Please send all correspondence here: nexus-feedback@sonatype.com⁶²⁸

What's New?

As we mentioned, if this is your first time trying a Nexus 3 milestone release, you should check out our notes for [Milestone 1\(see page 133\)](#). That provides a good background for many of the changes you can expect, especially those related to the user interface.

This release builds upon that functionality by offering the expanded features of Nexus Pro. That means a few things might not be exactly where you expect them... which is a good thing.

That's enough talking about it though. We know why you came, a list of new features in Nexus 3 Milestone 2. Keep reading to find out more.

New Default URL

The default URL for Nexus is `http://localhost:8081/`. Previously this had the context of `/nexus`. However, this has been removed.

Admin Area Changes

One of the core updates to the new user interface is the separation of administration and browsing functionality. Because of this, a few things were moved, and given their own first-class representation in the Administration area.

CLM



Product Licensing



Staging Admin



Staging Browse



PGP Configuration



⁶²⁸ mailto:nexus-feedback@sonatype.com

Enable User Token**View User Token**

Milestone 1 Release

This initial milestone build (M1) is a technical preview of things to come, and is part of a series of releases leading up to the final GA version of Nexus 3.0. The focus of M1 is the new user interface, though there are numerous refinements to the internals of Nexus not readily apparent in the new look and feel. Additionally, M1 is only being made available for Nexus OSS (M2, coming shortly, will cover Nexus Pro).

This milestone is being made available for anyone interested. We welcome feedback from the user community - See corresponding details below.

A word of CAUTION! before you begin

Some important things to keep in mind:

1. You should be experienced installing and administering Nexus already. If not, we recommend starting with the latest version of Nexus 2.xx.
2. It probably goes without saying, but do NOT upgrade a production server with any milestone builds.
3. Upgrades between milestone builds may NOT be fully supported
4. Fully updated documentation is not currently available, however we will be providing some items to help you get started. (See below)
5. Have fun, tell us what you like, what you don't, and if you are so inclined, try to break things.

What if I experience problems, or want to offer feedback?

Great, we'd love to hear from you. Please send all correspondence here: nexus-feedback@sonatype.com⁶²⁹.

So, what's New?

First and foremost, [Milestone 2](#)(see page 131) features support for Nexus Pro, as well as everything included in Milestone 1.

If you just joined us, Milestone 1 was the first release, and focused on changes which were mainly visual. Also, As we mentioned earlier, a number of improvements have been made under the hood. We won't go deep into the latter, since you won't see most of that.

However, a good place to start is by taking a look at the new console. After that, we'll move on to that very new look and feel, which you are likely to agree, is quite apparent. C'mon, let's take a look at this new technology.

⁶²⁹ mailto:nexus-feedback@sonatype.com

⚠ JSW (Java Service Wrapper) is no longer part of the distribution.

A brand new console / shell

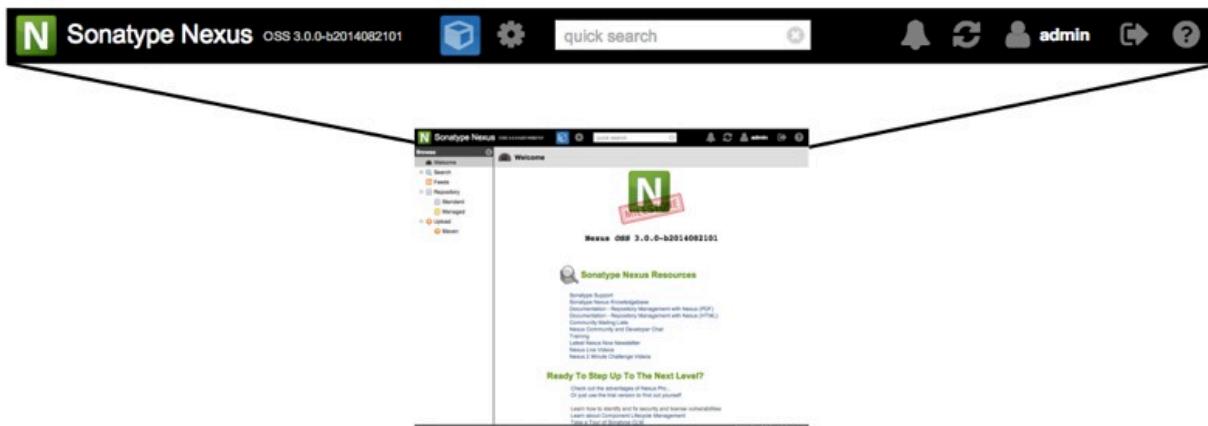
There is a new console, which is OSGI based on the Apache Karaf platform. For testing we recommend starting the repository manager in a terminal using

```
<install>/bin/nexus console
```

This will provide a view of the bootstrap logging (incase anything strange is going on). To exit the console use CTRL-D or 'shutdown' command.

The boot logs are located in `<install>/data/log/karaf.log`. See `<install>/bin/setenv` script to configure JAVA_HOME or other JVM options.

The navigation has changed, and there are three new buttons...



Just browsing

This button opens a panel on the left side of the screen. There you can search for components, review feeds, access repositories, and upload Maven components.



Admin on deck

Click this, and you'll see the administration functions are now in their own area. Just click this icon and you can manage everything from Repositories to Plugins. You may notice, we've added icons to make the administration area easier to navigate as well. Who am I? Clicking this button lets you manage account settings, like changing your password.

OK, technically there are seven...



Danger Will Robinson!

Nexus messages are a key piece of new functionality for Nexus, and in just a bit we'll show a quick video of just how they work. Click here to see what Nexus has been saying.



Suffering from stale data?

A “global refresh” replaces the seemingly random (and frequent!) placement previously. Now you can refresh the entire UI in a single click.



Coming and going?

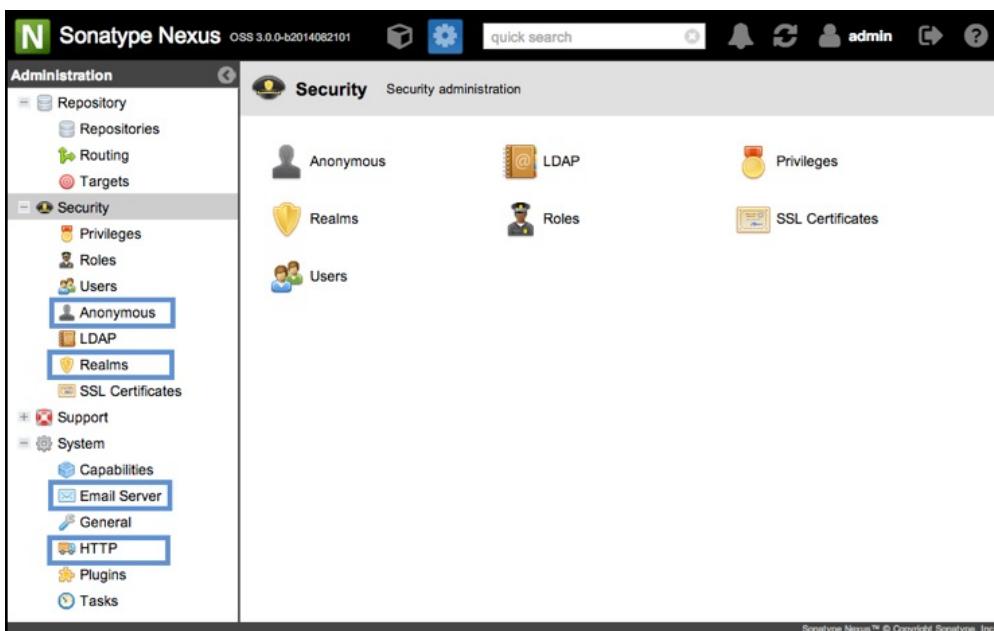
Click here to sign in, or out.



Need help?

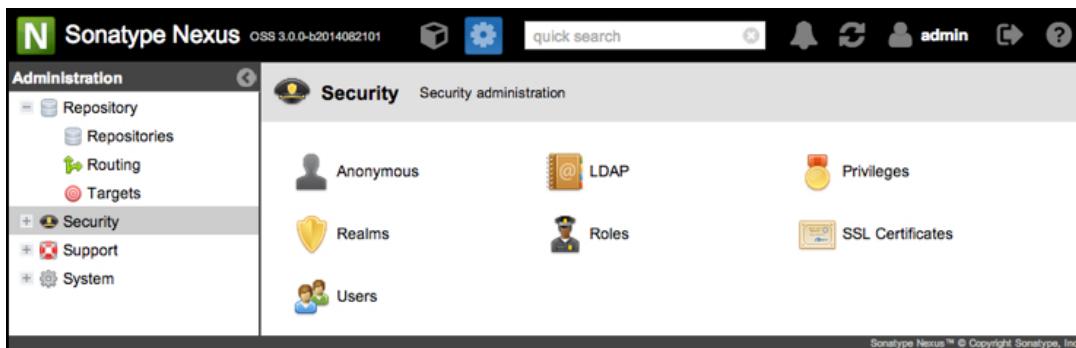
This icon is the place to start.

Navigating the Administration area has changed...



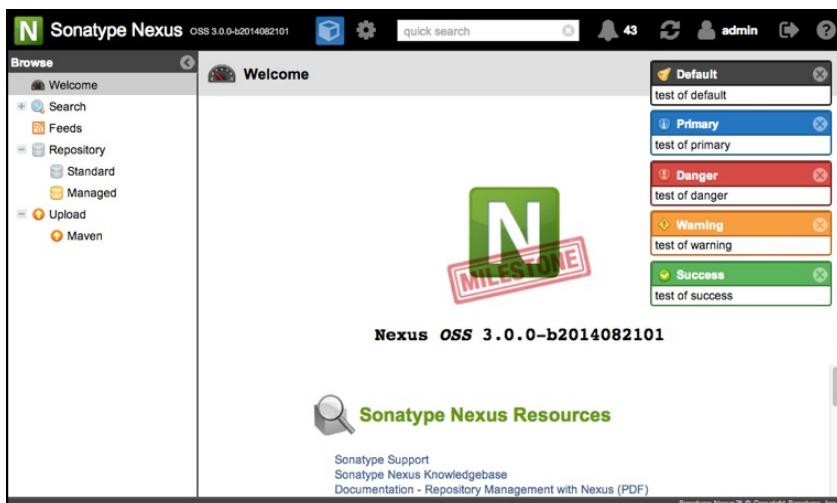
The Administration section has seen some notable refinements. We've abandoned things like the monolithic server configuration page. These options now have their own place within the appropriate administration category (see highlighted areas to the right).

... and there are a lot of new icons here as well.



When clicking on the various sections in the left navigation, the items in a particular category will display on the right (e.g. Security as shown) as well. Clicking on an icon will open that area.

Check out messages...

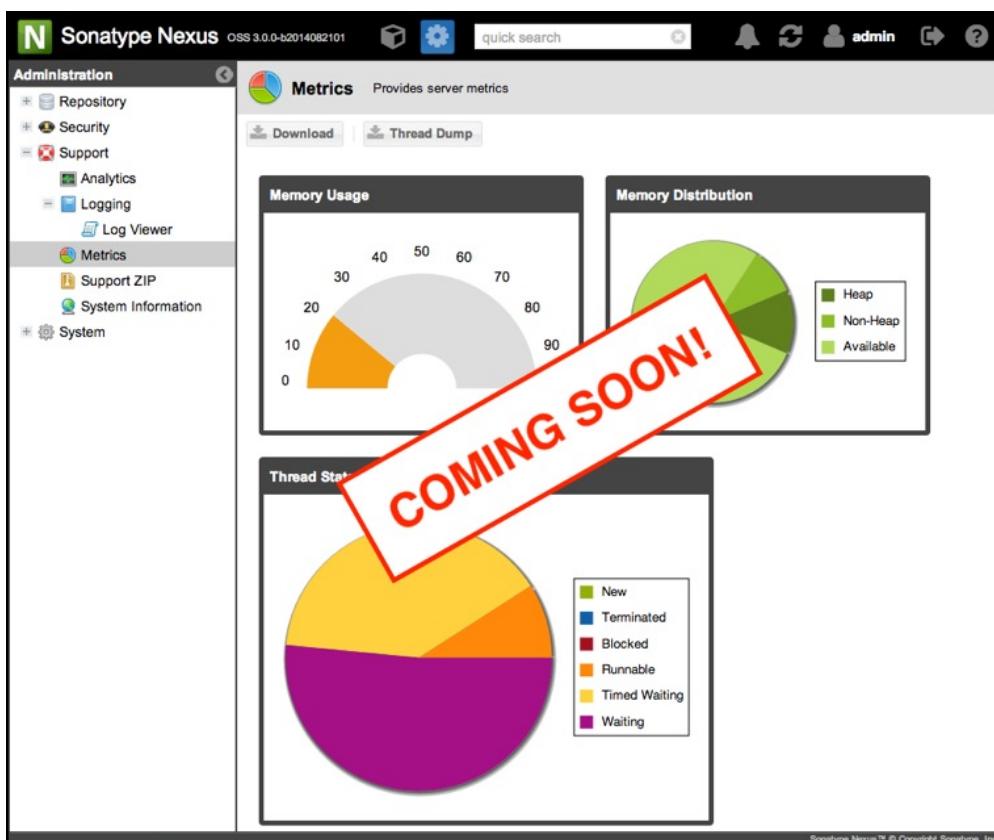


Messages provide a way to let you know that an action that has been taken was acknowledged by Nexus. For example, in the case of server-side validation if the content is not valid, it will report that back an appropriate error. When a message is delivered, it will be displayed for a few seconds before fading out. But, you have full control to clear and delete them as well. Just click the bell icon.

Plus, there is a whole lot more coming...

We'll let you know when M2 is ready. However, just so you are getting excited about it, here are some things you can expect.

- Updated UI for Nexus Pro
- Updated Documentation
- Bug fixes



System Requirements

Supported Versions

Sonatype fully supports versions of repository manager for one year after the release date. Older releases are supported on a best effort basis and the release dates are listed in our download archives. The terms of support are explained in section 3 of the [End User License Agreement](#)⁶³⁰.

Host Operating System

Any Windows, Linux or Macintosh operating system that can run a supported Java version will work. Other operating systems may work, but they are not tested by Sonatype.

The most widely used operating system for Nexus Repository Manager (NXRM) is Linux and therefore customers should consider it the best tested platform.

⁶³⁰ <https://www.sonatype.com/Usage/Software-Support-Policy>

Dedicated Operating System User Account

Unless you are just testing the repository manager or running it only for personal use, a dedicated operating system user account is strongly recommended to run each unique process on a given host.

 The NXRM process user is typically named 'nexus' and must be able to create a valid shell.

 **Important**

As a security precaution, do not run Nexus Repository Manager 3 as the root user.

Adequate File Handle Limits

NXRM3 will most likely want to consume more file handles than the per user default value allowed by your Linux or OSX operating system.

Running out of file descriptors can be disastrous and will most probably lead to data loss. Make sure to increase the limit on the number of open files descriptors for the user running Nexus Repository Manager permanently to 65,536 or higher prior to starting.

See <https://issues.sonatype.org/browse/NEXUS-12041> for additional background.

Linux

On most Linux systems, persistent limits can be set for a particular user by editing the `/etc/security/limits.conf` file. To set the maximum number of open files for both soft and hard limits for the `nexus` user to 65536, add the following line to the `/etc/security/limits.conf` file, where "nexus" should be replaced with the user ID that is being used to run the repository manager:

```
nexus - nofile 65536
```

This change will only take effect the next time the `nexus` process user opens a new session. Which essentially means that you will need to restart NXRM.

On Ubuntu systems there is a caveat: Ubuntu ignores the `/etc/security/limits.conf` file for processes started by `init.d`.

So if NXRM is started using `init.d` there, edit `/etc/pam.d/common-session` and uncomment the following line (remove the hash # and space at the beginning of the line):

```
# session    required    pam_limits.so
```

For more information refer to your specific operating system documentation.

If you're using `systemd` to launch the server the above won't work. Instead, modify the configuration file to add a `LimitNOFILE` line:

```
[Unit]
Description=nexus service
After=network.target

[Service]
Type=forking
LimitNOFILE=65536
ExecStart=/opt/nexus/bin/nexus start
ExecStop=/opt/nexus/bin/nexus stop
User=nexus
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

Mac OSX

The method to modify the file descriptor limits on OSX has changed a few times over the years. Please note your OS X version and follow the appropriate instructions.

For OS X Yosemite (10.10) and newer

1. Create the file: `/Library/LaunchDaemons/limit.maxfiles.plist`

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>limit.maxfiles</string>
    <key>ProgramArguments</key>
    <array>
        <string>launchctl</string>
        <string>limit</string>
        <string>maxfiles</string>
        <string>65536</string>
        <string>65536</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>ServiceIPC</key>
    <false/>
</dict>
</plist>
```

If this file already exists, then ensure the value is **at least 65536** as shown.

The file must be owned by `root:wheel` and have permissions `-rw-r--r--`.

```
sudo chmod 644 /Library/LaunchDaemons/limit.maxfiles.plist
sudo chown root:wheel /Library/LaunchDaemons/limit.maxfiles.plist
```

Reboot the operating system to activate the change.

2. Add a new line to

```
$install-dir/bin/nexus.vmoptions
containing:
```

```
-XX:-MaxFDLimit
```

Restart NXRM to activate the change.

For OS X Lion (10.7) up to OS X Mavericks (10.9)

1. Create and edit the system file `/etc/launchd.conf` using this command:

```
sudo sh -c 'echo "limit maxfiles 65536 65536" >> /etc/launchd.conf'
```

Reboot the operating system to activate the change.

2. Add a new line to `$install-dir/bin/nexus.vmoptions` containing:

```
-XX:-MaxFDLimit
```

Restart NXRM to activate the change.

Windows

Windows operating systems do not need file handle limit adjustments.

Docker

The Nexus Repository Docker images are configured with adequate file limits. Some container platforms such as Amazon ECS will override the default limits. On these platforms it is recommended that the Docker image be run with the following flags:

```
--ulimit nofile=65536:65536
```

Java

Nexus Repository Manager 3 is a Java server application that requires a **Java 8 SE standard compatible runtime**.

Java runtime versions other than 8 are not supported - do not use them.

Only 64 bit Java is supported, do not use 32 bit Java.

Windows, macOS and Docker image distributions include a bundled JRE. Linux distributions require a JRE or JDK to be installed.

Java Development Kit (JDK) installs are recommended because of their included diagnostic tools. Java Runtime Environment (JRE) installs will work, but lack useful tools like jstack.

As of repository manager version 3.15.0, product releases are tested against a build of OpenJDK which is fully compatible to the Java SE 8 standard. Prior repository manager versions were tested against the official Oracle JDK distribution.

Any fully compatible Java SE 8 standard runtime distribution *should* work.

Users have reported that OpenJDK builds have worked against repository manager versions older than 3.15.0, but Sonatype was not actively testing this.

Sonatype makes no claim that *all* JDK distributions and operating system combinations will work. The IBM JDK can be used on platforms where there is no other option, but our support of these other platforms is done on a best effort basis.

GNU GCJ will not work and is not supported.

CPU

Performance is primarily bounded by IO (disk and network) rather than CPU. Available CPUs will impact longer running operations and also the [thread allocation algorithms of the web container](#)⁶³¹.

Minimum CPUs: 4

Recommended CPUs: 8+

Memory

Configurable Memory Types

Visit the [Configuring the Runtime Environment](#)(see page 0) page to learn how to change the default memory settings.

JVM Heap Memory

Heap memory stores runtime application objects. A min (-Xms) and max (-Xmx) value must be specified and the values should be identical.

Increasing the heap memory larger than recommendations or setting the min and max values to be different is **not recommended**. This will create performance issues causing the operating system to thrash needlessly.

Unless you have evidence that a max heap of 4GB is consistently utilized or there are frequent lengthy garbage collection pauses that cannot be explained by software bugs, then **do not set max heap size larger than 4GB**.

JVM Direct Memory

Direct memory is allocated outside of and distinct from heap memory. A max value must be configured.

⁶³¹ <https://support.sonatype.com/hc/en-us/articles/360000744687-Understanding-Eclipse-Jetty-9-4-8-Thread-Allocation>

Host Physical Memory

The total memory allocated to the entire operating system or virtual hardware, commonly referred to as RAM.

Memory Requirements

The requirements assume there are no other significant memory hungry processes running on the same host.

	JVM Heap	JVM Direct	Host Physical/RAM
Minimum (default)	2703MB	2703MB	8GB
Maximum	4GB	(host physical/RAM * 2/3) - JVM max heap	<i>no limit</i>

General Memory Guidelines

- minimum physical/RAM memory on the host 8GB
- minimum heap (-Xms) must equal set maximum heap (-Xmx)
- minimum heap size 2703MB
- maximum heap size <= 4GB
- minimum direct memory (-XX:MaxDirectMemorySize) size 2703MB
- minimum unallocated host physical/RAM memory should be no less than 1/3 of total physical RAM to allow for virtual memory swap
- max heap + max direct memory <= host physical/RAM * 2/3

Instance Memory Sizing Profiles

These profiles help gauge the typical physical memory requirements needed for a dedicated server host running repository manager. *Due to the inherent complexities of use cases, one size does not fit all and this should only be interpreted as a guideline.*

Profile Use Case	Physical/RAM Memory
small, personal repositories < 20 total blobstore size < 20GB single repository format type	8GB minimum

Profile Use Case	Physical/RAM Memory
medium, team repositories < 50 total blobstore size < 200GB a few repository formats	16GB
large, enterprise repositories > 50 total blobstore size > 200GB diverse set of repository formats	32GB+

Example Maximum Memory Configurations

Physical/RAM Memory	Example Maximum Memory Configuration
8GB	<code>-Xms2703M -Xmx2703M -XX:MaxDirectMemorySize=2703M</code>
12GB	<code>-Xms4G -Xmx4G -XX:MaxDirectMemorySize=4014M</code>
16GB	<code>-Xms4G -Xmx4G -XX:MaxDirectMemorySize=6717M</code>
32GB	<code>-Xms4G -Xmx4G -XX:MaxDirectMemorySize=17530M</code>

Physical/RAM Memory	Example Maximum Memory Configuration
64GB	<code>-Xms4G -Xmx4G -XX:MaxDirectMemorySize=39158M</code>

Advanced Database Memory Tuning

Refer to another article which outlines [additional memory tuning procedures](#)⁶³².

Temporary Directory

The [temporary directory](#)(see page 189) at

`$data-dir/tmp`

must not be mounted with noexec or repository manager startup will fail

with `java.lang.UnsatisfiedLinkError` message of failed to map segment from shared object:
Operation not permitted .

Disk Space

Application Directory - The size of this directory varies slightly each release. It currently around 330 MB. It is normal to have multiple application directories installed on the same host over time as repository manager is upgraded.

Data Directory - On first start, repository manager creates the base files needed to operate. The bulk of disk space will be held by your deployed and proxied artifacts, as well as any search indexes. This is highly installation specific, and will be dependent on the repository formats used, the number of artifacts stored, the size of your teams and projects, etc. It's best to plan for a lot though, formats like Docker and Maven can use very large amounts of storage (500Gb easily).

File Systems to avoid

File systems known to be unreliable are:

1. glusterfs (split-brain and slow performance)
2. FUSE based user space filesystems

Additionally, we strongly recommend to avoid using NFS, Amazon EFS and SMB for anything other than blob storage in Nexus Repository 3.x, especially in large installations and high availability setups, as this can cause severe performance degradation. If NFS is used for blob storage we recommend to only use NFSv4; NFSv3 is known to provide inadequate performance. We also have some [optimization suggestions to use at](#)

⁶³² <https://support.sonatype.com/hc/en-us/articles/115007093447>

your discretion⁶³³. Also consider the `noatime` option for your Nexus Repository work directory mounts and limit the symbolic links used as this will cause increased overhead whenever paths need to be resolved to an absolute file path.

Web Browser

Our general policy is to support the most recent modern browser version for your supported OS at time of NXRM release date.

Vendor	Browser	Versions
Google	Chrome	latest at NXRM release
Mozilla	Firefox	latest and ESR ⁶³⁴ at NXRM release
Apple	Safari	latest at NXRM release
Microsoft	Internet Explorer	latest (IE11) at NXRM release
Microsoft	Edge	latest at NXRM release

Upgrade Compatibility - Repository Manager 2 to 3

Requirement: Always Use Latest Versions

Sonatype always recommends having the latest version 2 upgrading to the latest version 3 (as well as IQ Server for applicable users). Latest versions have the latest bug fixes, especially important for the upgrade process and compatibility with new features of Nexus Repository Manager 3.

Upgrading to an older NXRM3 then subsequently updating to the latest NXRM3 is not equivalent, as it will hit old upgrade bugs.

Requirement: Always Use Same License

When upgrading from repository manager 2 to 3, both versions must have the same licensing setup.

The same license file can be used in NXRM2 and NXRM3 or a license can always be applied at a later time.

⁶³³ <https://support.sonatype.com/hc/en-us/articles/213465258-Optimizing-Nexus-Disk-IO-Performance>

⁶³⁴ <https://www.mozilla.org/en-US/firefox/organizations/>

Licensing: Valid Upgrade Path	Licensing: Unsupported Upgrade Path
Nexus Repository Manager OSS 2 → Nexus Repository Manager OSS 3	Nexus Repository Manager OSS 2 → Nexus Repository Manager Pro 3
Nexus Repository Manager Pro 2 → Nexus Repository Manager Pro 3	Nexus Repository Manager Pro 2 → Nexus Repository Manager OSS 3

Supported Nexus Repository Manager Upgrade Paths

Find the repository manager version you have in the left column. Follow the upgrade procedure in the right column.

Source Nexus Repository Manager	Required Upgrade Procedure
3.1.0 and newer	1. Upgrade to latest available 3.x version
3.0.0 to 3.0.2	1. Learn about important directory layout changes introduced in 3.1.0 ⁶³⁵ 2. Upgrade to latest available 3.x version
3 milestone 7	1. Upgrade to 3.0.0 ⁶³⁶ 2. Learn about important directory layout changes introduced in 3.1.0 ⁶³⁷ 3. Upgrade to latest available 3.x version
3 milestone 1 to milestone 6	Upgrading these versions is not supported .
Latest 2.x version	1. Upgrade to latest available 3.x version
2.0 or any later 2.x version	1. Upgrade to latest available 2.x version 2. Upgrade to latest available 3.x version
1.x	1. Upgrade to any 2.x version up to 2.7.2 ¹ 2. Upgrade to latest available 2.x version 3. Upgrade to latest available 3.x version

⁶³⁵ <https://support.sonatype.com/hc/en-us/articles/231723267>

⁶³⁶ <https://support.sonatype.com/hc/en-us/articles/222159808-How-to-Upgrade-Repository-Manager-3-0-0-m7-to-3-0-0-final>

⁶³⁷ <https://support.sonatype.com/hc/en-us/articles/231723267>

1 Versions after 2.7.2 do not include support for upgrading 1.x versions

The latest Nexus Repository Manager version 2 can be downloaded [here](#)⁶³⁸.

The latest Nexus Repository Manager version 3 can be downloaded [here](#)⁶³⁹.

Minimally Compatible Server Versions for Lifecycle Features

Below is a matrix of Sonatype server product versions minimally verified for lifecycle features. This does not mean all features are supported by these versions.

The table is suggesting that major issues should not be encountered running these minimal versions congruently.

If you encounter problems with any of the versions listed, we advise upgrading to the most recent version first to attempt resolution of the issue.

Version 2	Version 3	Minimum Lifecycle Firewall compatible IQ Server
2.14.8	3.12.1	1.46.0
2.14.6	3.8.0	1.42.0
2.14.5	3.7.1	1.33.0
2.14.4	3.4.0	Not Supported
2.14.3	3.2.1	Not Supported
2.14.2	3.2	Not Supported
2.14.1	3.1	Not Supported

Repository Manager 2 to 3 Feature Equivalency

Nexus Repository Manager 3 (available via the Nexus Repository 3 OSS or Pro Product license) represents a complete rewrite of architecture and functionality. As of version 3.1.x many of these features have a new, modern equivalency. However, there is not yet 1:1 parity with regard to features, and in some cases, that may be the case for some time. Please use the matrix below to plan any upgrades and/or installations.

⁶³⁸ <http://links.sonatype.com/products/nxrm2/download>

⁶³⁹ <http://links.sonatype.com/products/nxrm3/download>

Repository 2 Feature	Repository 3 Equivalent	Note
Server Settings	👍	These are settings under Administration -> Server in Nexus Repository Manager 2
Security	👍	Please refer to the Security (see page 309) section
Local User Accounts	👍	
Privileges	👍	Names may change on upgrade from Nexus Repository Manager 2 to 3
Roles	👍	
Maven Repositories	👍	
Proxy Repository of maven.oracle.com	👍	See KB article to help getting this configured ⁶⁴⁰
Browse Remote	✖	
Repository Targets	👍	Maps to content selectors (see page 251)
Manual Routing Rules	✖	
Maven Settings	✖	We anticipate implementing an alternate feature that is applicable to any supported repository format.
Automatic Routing	✖	We will consider similar optimization features like this in the future as warranted.

⁶⁴⁰ <https://support.sonatype.com/hc/en-us/articles/213465728>

Repository 2 Feature	Repository 3 Equivalent	Note
Smart Proxy	✖	We are planning a set of instance-to-instance replication features.
Procurement	👍	The Procurement feature, which is limited to Maven in Nexus Repository 2, has been deprecated. Going forward, Sonatype's Nexus Firewall ⁶⁴¹ solution provides a far more robust and comprehensive set of features aimed at managing risky components via your repository.
Staging	👍	Nexus Repository Manager 2.x Staging has been replaced with a more robust solution that relies on component tagging to manage workflow. See here (see page 352) for details.
Custom Metadata	👍	Nexus Repository Manager 3 has the ability to associate arbitrary data with any component in any repository format via it's tagging (see page 357) feature. However, there is currently no upgrade mechanism which will preserve existing custom metadata from Nexus Repository Manager 2.
YUM Repositories	👍	YUM (see page 460) proxy support is included in the Nexus Repository Manager 3.5.0 release and hosted support is included in the Nexus Repository Manager 3.8.0 release. Feature was completed in 3.11.0.
p2 and p2 Update Site Repositories	👍	
RubyGems Repositories	👍	

⁶⁴¹ <https://www.sonatype.com/nexus-firewall>

Repository 2 Feature	Repository 3 Equivalent	Note
PyPI Repositories	👍	
npm Repositories	👍	
NuGet Repositories	👍	
NuGet API Keys	👍	
Bower Repositories	👍	
Site Repositories	👍	This is called Raw repository format in Nexus Repository Manager 3
System Feeds	✖	We have no plans to upgrade System Feeds as the data is specific to the instance where it was generated. Our roadmap includes improving auditing capabilities and adding webhooks into Nexus Repository Manager 3.1.
Branding	👍	The Nexus Repository Manager 2 branding configuration will not be upgraded automatically. Nexus Repository Manager 3 uses the <i>UI: Branding</i> capability.
REST API	👍	A Scripting API is available which may be suitable for some use cases, however, the Nexus Repository Manager 2 REST APIs are not compatible with Nexus Repository Manager 3. Our roadmap includes adding various REST API endpoints at a later date.

Repository 2 Feature	Repository 3 Equivalent	Note
Non-Sonatype Repository Manager Plugins	✖	Nexus Repository Manager 2 plugins are not compatible with Nexus Repository Manager 3 OSGI bundle architecture
SSL Certificates	👍	SSL certificates trusted in the UI will be upgraded, but not all of them may be applicable to a new server. Trusted certificates should be reviewed after your upgrade. Upgrade does not include the certificate that Nexus Repository Manager uses for its own HTTPS connection, if present.
Analytics	✖	This little-used feature was removed in 3.15.
LDAP	👍	
Nexus Repository OSS Servers	👍	
Nexus Repository Pro (Enterprise) Servers	👍	Individual LDAP Server configurations will be upgraded. Backup Mirror servers will not be upgraded.
Mapped Users and Roles	👍	
Atlassian Crowd	👍	After your upgrade, you will need to ensure your Crowd application is configured to accept connections from Nexus Repository Manager 3
Mapped User and Roles	👍	

Repository 2 Feature	Repository 3 Equivalent	Note
User Token	👍	See section on User Token Upgrade (see page 201)
IQ Server Configuration	👍	
Repository Health Check Analysis (RHC)	👍	
RUT Auth Realm	👍	This will NOT be upgraded automatically. This feature can be a security concern when enabled. Configure this manually inside Nexus Repository Manager 3 when you are confident that Nexus Repository Manager 3 access is protected.

Repository Manager Feature Matrix

The table below provides detail for the various the complete range of Repository Manager features, as well as the corresponding solution (license) that provides access to them. Whether you're looking to upgrade, curious about a specific format, or just simply want a bird's-eye view of features, check back often as we update this table whenever new features are released. If you are an existing user of Nexus Repository Manager 2.x and are looking to understand the equivalent feature in 3.x, check out our [Feature Equivalency article](#)(see page 148).

Feature	2 OSS	2 Pro	3 OSS	3 Pro
Bower	✗	✗	👍	👍
Docker	✗	✗	👍	👍
Git Large File Storage (LFS)	✗	✗	👍	👍
Go Modules	✗	✗	👍	👍
Maven	👍	👍	👍	👍
npm	limited	limited	👍	👍
APT	✗	✗	👍	👍
R	✗	✗	👍	👍

Feature	2 OSS	2 Pro	3 OSS	3 Pro
CocoaPods	✗	✗	👍	👍
Conda	✗	✗	👍	👍
NuGet	👍	👍	👍	👍
OBR	✗	👍	See Below(see page 155)	See Below(s ee page 155)
P2	✗	👍	👍	👍
PyPI	✗	✗	👍	👍
RubyGems	👍	👍	👍	👍
YUM	👍	👍	👍	👍
Gradle	👍	👍	👍	👍
Smart Proxy	✗	👍	See Below(see page 155)	See Below(s ee page 155)
Upgrading 2x to 3x	✗	✗	👍	👍
Unlimited Deployment	👍	👍	👍	👍
Component Search	limited	👍	👍	👍
Upload Third Party Artifacts via UI	limited	limited	👍	👍
RHC	👍	👍	👍	👍
Custom Metadata	✗	👍	See Below(see page 155)	See Below(s ee page 155)
Improved Backup & Restore	✗	✗	👍	👍
Provisioning API	✗	✗	👍	👍
REST	👍	👍	👍	👍
Plugins	👍	👍	See Below(see page 155)	See Below(s ee page 155)
Open Source Integration	👍	👍	👍	👍
Remote Auth Token (RUT)	✗	👍	👍	👍
Custom Access Controls	👍	👍	👍	👍
Repo Targets / Content Selectors	👍	👍	👍	👍

Feature	2 OSS	2 Pro	3 OSS	3 Pro
Enterprise LDAP	👍	👍	👍	👍
Crowd	✗	👍	✗	👍
SAML	✗	✗	✗	👍
Staging & Build Promotion	✗	👍	✗	👍
Community Support	👍	👍	👍	👍
Enterprise Support	✗	👍	✗	👍
User Token Support	✗	👍	✗	👍

If you have questions about a specific feature, contact nexus-feedback@sonatype.com⁶⁴².

OBR

We aim to provide support for every format available. However, we also have to prioritize according to the needs of our customers. If you want to see OBR supported, check out the Nexus Repository Manager JIRA project and add a [comment with your support](#)⁶⁴³.

Replication

In an upcoming version of Nexus Repository Manager Pro, we'll be introducing a distributed replication approach that allows for simplified replication of content between repository manager instances.

Custom Metadata

The Custom Metadata feature has been replaced with the new [component tagging](#)(see page 357) feature.

Plugins

There are are number of plugins available for NXRM3, and more coming all the time. View the entire list of plugins at [Nexus Exchange](#)⁶⁴⁴. If you've created a Nexus Repository Manager plugin, reach out to our [Community Advocate](#)⁶⁴⁵, and we'll help you through the process of getting it out to other Nexus Repository users.

⁶⁴² mailto:nexus-feedback@sonatype.com

⁶⁴³ <https://issues.sonatype.org/browse/NEXUS-12349>

⁶⁴⁴ <http://exchange.sonatype.com/>

⁶⁴⁵ mailto:community@sonatype.com

Repository Manager Pro Features

Atlassian Crowd Support

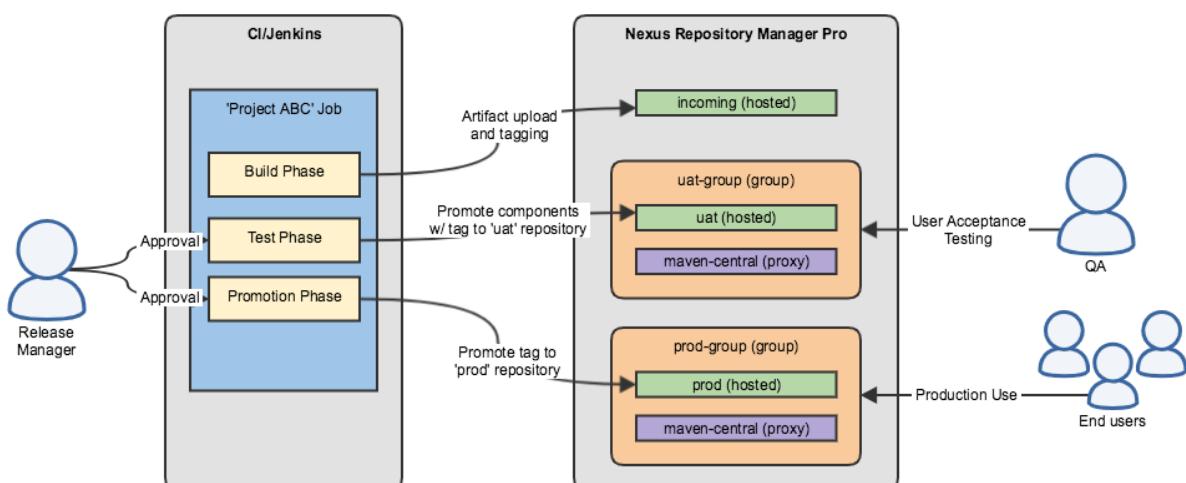
Atlassian Crowd is a single sign-on and identity management product that many organizations use to consolidate user accounts and control which users and groups have access to which applications. Atlassian Crowd support is a feature preinstalled and ready to configure in Nexus Repository Manager Pro. Nexus Repository Manager contains a security realm that allows you to configure the repository manager to authenticate against an Atlassian Crowd instance.

Staging and Build Promotion

In modern software development, it is imperative to thoroughly test software before it is deployed to a production system or externally accessible repository. Mostly commonly a release candidate will first be deployed to a staging system which is a close copy of the production system so it can undergo a series of rigorous tests before a decision is made to promote it to production or return it to development.

The staging functionality in Nexus Repository Manager Pro supports promotion of software components matching your organization's software development life cycle phases by moving those components between repositories. This allows the creation of isolated release candidates that can be discarded or promoted to make it possible to support the decisions that go into certifying a release.

Typical Staging Workflow



Tagging

Tagging provides the ability to mark a set of components with a tag so they can be logically associated to each other. The usage of the tags is up to you but the most common scenarios would be a CI build ID for a project (e.g. project-abc-build-142) or a higher level release train when you are coordinating multiple projects together as a single unit (e.g. release-train-13). Tagging is used extensively by the [Staging\(see page 352\)](#) feature.

User Token Support

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 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.

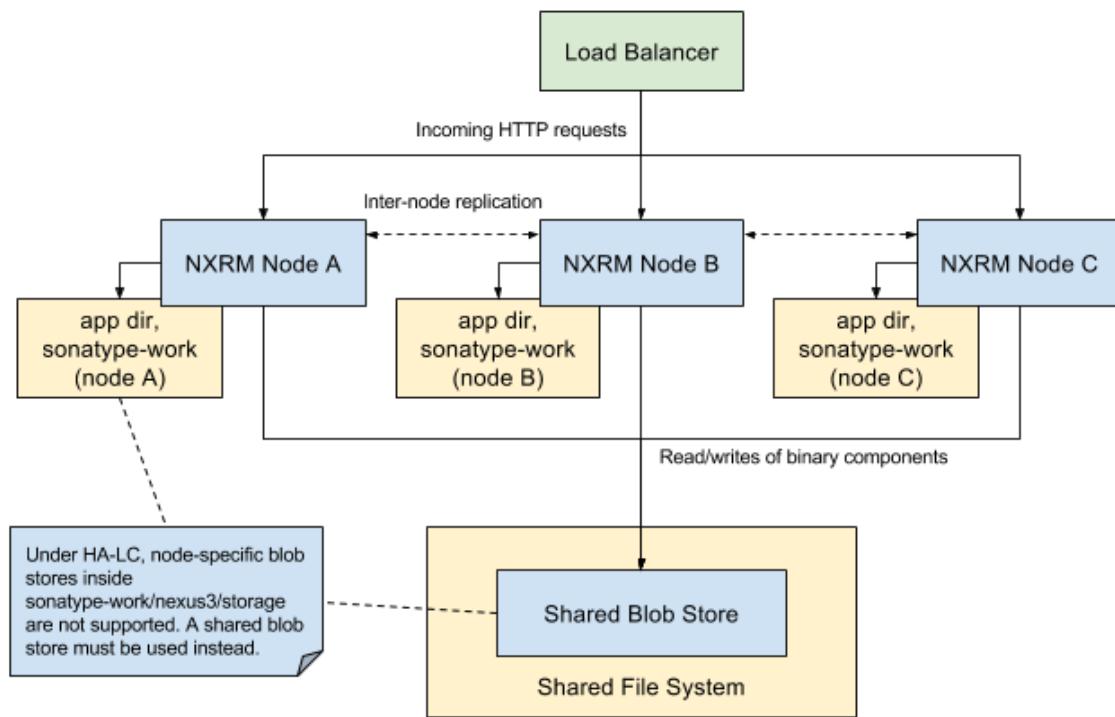
Other build systems use similar approaches and can benefit from the usage of user tokens as well. Nexus Repository Manager Pro's user token feature establishes a two-part token for the user. Usage of the token acts as a substitute method for authentication that would normally require passing your username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

High Availability

High Availability Clustering (HA-C) is a feature designed to improve uptime by having a cluster of redundant NXRM "nodes" (instances) within a single data center. This feature allows you to maintain availability to your Nexus Repository Manager in the event one of the nodes becomes unavailable.

Typical HA Configuration



Repository Health Check

Nexus Repository users can now automatically identify open source security risks at the earliest stages of their DevOps pipeline. Specifically, the RHC feature empowers software development teams with three important capabilities:

- Provides actionable guidance on which components housed in the repository manager should be upgraded or replaced.
- Prioritizes the list of vulnerable components by severity and impact, detailing how many times each component was downloaded from the repository manager by developers in the past 30 days.
- Reveals month-over-month metrics on the hygiene of the organization’s software supply chain to identify improving standards or worrisome trends.

Example Health Check Overview



Customer Success

Sonatype significantly invests in each customer through our Customer Success team. This team is unique in that they will proactively and continuously collaborate with you to ensure you derive ongoing value from your Sonatype investment.

To start, they always recommend customers invest in workshops and training, which they deliver. These accelerate time-to-value and provide a solid foundation for long-term success. What's more, at no additional cost, they will coach you from the beginning. They will create and collaborate with you on a joint roll-out plan. They will provide you with best-practices, curate relevant materials for you, help you with designing architecture, and continue to monitor your progress towards your goals. And they will do this throughout your lifecycle as a Sonatype customer.

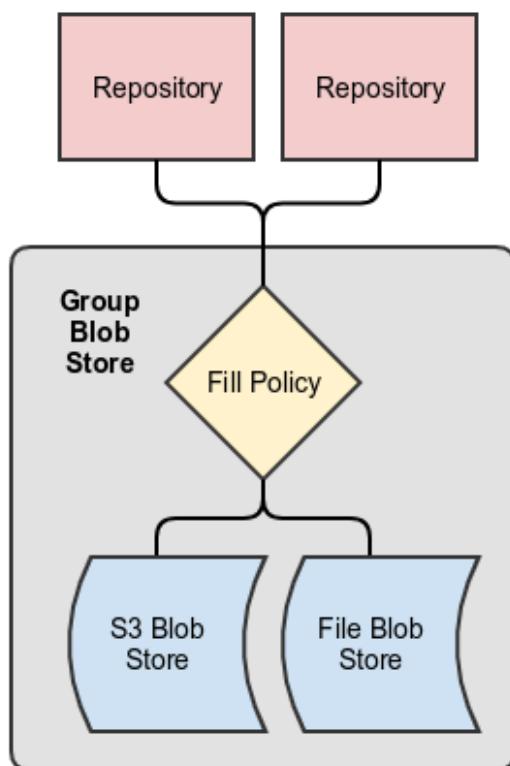
Enterprise Support

Nexus Repository Pro users also get access to our World Class Support organization. This highly skilled group will work with you to resolve issues encountered while using the product. Detailed support information can be found at <https://www.sonatype.com/usage/software-support-policy>.

Group Blob Stores

A group blob store combines multiple blob stores together so that they act as a single blob store for repositories. Members of the group can be added or removed. Fill Policies are selectable algorithms that determine to which group member a blob is written. These features significantly increase the flexibility customers have in using and upgrading their storage. More information can be found in the [Storage Guide](#)(see page 281).

Example Group Blob Store



SAML Authentication and Single Sign-On

Nexus Repository Pro integrates with SAML Identity Providers to allow identity, authentication, and authorization to be managed centrally. Using SAML, NXRM acts as service provider which receives users' authentication and authorization information from external Identity Providers. Administrators can manage users and roles in one place, and users can sign in with Single Sign-On credentials.

Repository Manager Concepts

Using the Nexus Repository Manager as well as the tools for software supply chain automation using the Nexus IQ Server and associated tools of the Nexus platform requires an understanding of a few concepts and terms like component, repository and repository format. This section provides you with all the necessary background and knowledge as well as an idea of a progression in your usage of the tools from the Nexus platform.

Topics in this section:

- [Components, Repositories, and Repository Formats](#)(see page 161)
- [An Example - Maven Repository Format](#)(see page 163)
- [Managing Repositories](#)(see page 167)

- [Software Supply Chain Automation](#)(see page 168)

Components, Repositories, and Repository Formats

The Nexus platform, with Nexus Repository Manager Pro, Nexus Repository Manager OSS and Nexus IQ Server, is all about working with components and repositories.

Components

A component is a resource like a library or a framework that is used as part of your software application at run-time, integration or unit test execution time, or required as part of your build process. It could be an entire application or a static resource like an image. Typically these components are archives of a large variety of files including:

- Java byte code in class files
- C object files
- text files e.g. properties files, XML files, JavaScript code, HTML, CSS
- binary files such as images, PDF files, sound files

The archives are using numerous formats such as

- Java JAR, WAR, EAR formats
- plain ZIP or .tar.gz files
- other package formats such as NuGet packages, Ruby gems, NPM packages
- executable formats such as .exe or .sh files, Android APK files, various installer formats

Components can be composed of multiple, nested components themselves. For example, consider a Java web application packaged as a WAR component. It contains a number of JAR components and a number of JavaScript libraries. All of these are standalone components in other contexts and happen to be included as part of the WAR component.

Components provide all the building blocks and features that allow a development team to create powerful applications by assembling them and adding their own business related components to create a full-fledged, powerful application.

In different tool-chains components are called artifact, package, bundle, archive and other terms. The concept and idea remains the same and component is used as the generic term.

Components are identified by a set of specific values - the coordinates. A generic set of these coordinates is the usage of group, name and version. The names and the usage of these coordinates changes with the tool-chains used. Components can also be the anchor for further metadata.

Assets

Assets are the material addition to all this metadata. The actual archive file is an asset associated with the component. Many formats have a one-to-one mapping for component to asset.

More complex formats however have numerous assets associated with a component. For example, a typical JAR component in a Maven repository is defined at least by the POM and the JAR files - both of which constitute separate assets belonging to the same components. Additional files such as JavaDoc or Sources JAR files are assets that belong to the same component.

The Docker format, on the other hand, gives assets unique identifiers and calls them Docker layers. These assets can be reused for different components - the Docker images. A Docker layer, for example, could be a specific operating system referenced by multiple Docker images.

Components in Repositories

A wide variety of components exists and more are continuously created by the open source community as well as proprietary vendors. There are libraries and frameworks written in various languages on different platforms that are used for application development every day. It has become a default pattern to build applications by combining the features of multiple components with your own custom components containing your application code to create an application for a specific domain.

In order to ease the consumption and usage of components, they are aggregated into collections of components. These are called a repository and are typically available on the internet as a service. On different platforms terms such as registry and others are used for the same concept.

Examples for such repositories are:

- Central Repository, also known as Maven Central
- NuGet Gallery
- RubyGems.org⁶⁴⁶
- npmjs.org⁶⁴⁷

Components in these repositories are accessed by numerous tools including:

- package managers like npm, nuget, gem
- build tools such as Maven, Gradle, rake, grunt
- IDE's such as Eclipse, IntelliJ, Visual Studio

⁶⁴⁶ <http://RubyGems.org>

⁶⁴⁷ <http://npmjs.org>

Repository Formats

The different repositories use different technologies to store and expose the components in them to client tools. This defines a repository format and as such is closely related to the tools interacting with the repository.

For example, the Maven repository format relies on a specific directory structure defined by the identifiers of the components and a number of XML formatted files for metadata. Component interaction is performed via plain HTTP commands and some additional custom interaction with the XML files.

Other repository formats use databases for storage and REST API interactions, or different directory structures with format specific files for the metadata.

An Example - Maven Repository Format

Maven developers are familiar with the concept of a repository, since repositories are used by default. The primary type of a binary component in a Maven format repository is a JAR file containing Java byte-code. This is due to the Java background of Maven and the fact that the default component type is a JAR. Practically however, there is no limit to what type of component can be stored in a Maven repository. For example, you can easily deploy WAR or EAR files, source archives, Flash libraries and applications, Android archives or applications or Ruby libraries to a Maven repository.

Every software component is described by an XML document called a Project Object Model (POM). This POM contains information that describes a project and lists a project's dependencies – the binary software components, which a given component depends upon for successful compilation or execution.

When Maven downloads a component like a dependency or a plugin from a repository, it also downloads that component's POM. Given a component's POM, Maven can then download any other components that are required by that component.

Maven and other tools, such as Ivy or Gradle, which interact with a Maven repository to search for binary software components, model the projects they manage and retrieve software components on-demand from a repository.

The Central Repository

When you download and install Maven without any customization, it retrieves components from the Central Repository. It serves millions of Maven users every single day. It is the default, built-in repository using the Maven repository format and is managed by Sonatype. Statistics about the size of the Central Repository are available at <https://search.maven.org/stats>.

The Central Repository is the largest repository for Java-based components. It can be easily used from other build tools as well. You can look at the Central Repository as an example of how Maven repositories operate

and how they are assembled. Here are some of the properties of release repositories such as the Central Repository:

Component Metadata

All software components added to the Central Repository require proper metadata, including a Project Object Model (POM) for each component that describes the component itself and any dependencies that software component might have.

Release Stability

Once published to the Central Repository, a component and the metadata describing that component never change. This property of a release repository, like the Central Repository, guarantees that projects that depend on releases will be repeatable and stable over time. While new software components are being published every day, once a component is assigned a release number on the Central Repository, there is a strict policy against modifying the contents of a software component after a release.

Component Security

The Central Repository contains cryptographic hashes and PGP signatures that can be used to verify the authenticity and integrity of software components served and supports connections in a secure manner via HTTPS.

Performance

The Central Repository is exposed to the users globally via a high performance content delivery network of servers.

In addition to the Central Repository, there are a number of major organizations, such as Red Hat, Oracle or the Apache Software foundation, which maintain separate additional repositories. Best practice to facilitate these available repositories is to install Nexus Repository Manager OSS or Nexus Repository Manager Pro and use it to proxy and cache the contents on your own network.

Component Coordinates and the Repository Format

Component coordinates create a unique identifier for a component. Maven coordinates use the following values: groupId, artifactId, version, and packaging. This set of coordinates is often referred to as a GAV coordinate, which is short for Group, Artifact, Version coordinate. The GAV coordinate standard is the foundation for Maven's ability to manage dependencies. Four elements of this coordinate system are described below:

groupId

A group identifier groups a set of components into a logical group. Groups are often designed to reflect the organization under which a particular software component is being produced. For example, software components being produced by the Maven project at the Apache Software Foundation are available under the groupId `org.apache.maven`.

artifactId

An artifactId is an identifier for a software component and should be a descriptive name. The combination of groupId and artifactId must be unique for a specific project.

version

The version of a project ideally follows the established convention of [semantic versioning](#)⁶⁴⁸. For example, if your simple-library component has a major release version of 1, a minor release version of 2 and point release version of 3, your version would be `1.2.3`. Versions can also have alphanumeric qualifiers which are often used to denote release status. An example of such a qualifier would be a version like "`1.2.3-BETA`" where BETA signals a stage of testing meaningful to consumers of a software component.

packaging

Maven was initially created to handle JAR files, but a Maven repository is completely agnostic about the type of component it is managing. Packaging can be anything that describes any binary software format including: zip, nar, war, ear, sar and aar.

Tools designed to interact Maven repositories translate component coordinates into a URL which corresponds to a location in a Maven repository. If a tool such as Maven is looking for version `1.2.0` of the commons-lang JAR in the group `org.apache.commons`, this request is translated into:

```
<repoURL>/org/apache/commons/commons-lang/1.2.0/commons-lang-1.2.0.jar
```

Maven also downloads the corresponding POM for commons-lang 1.2.0 from:

```
<repoURL>/org/apache/commons/commons-lang/1.2.0/commons-lang-1.2.0.pom
```

This POM may contain references to other components, which are then retrieved from the same repository using the same URL patterns.

⁶⁴⁸ <http://semver.org/>

Release and Snapshot Repositories

A Maven repository stores two types of components: releases and snapshots. Release repositories are for stable, static release components. Snapshot repositories are frequently updated repositories that store binary software components from projects under constant development.

While it is possible to create a repository which serves both release and snapshot components, repositories are usually segmented into release or snapshot repositories serving different consumers and maintaining different standards and procedures for deploying components. Much like the difference between networks, a release repository is considered like a production network and a snapshot repository is more like a development or a testing network. While there is a higher level of procedure and ceremony associated with deploying to a release repository, snapshot components can be deployed and changed frequently without regard for stability and repeatability concerns.

The two types of components managed by a repository manager are:

Release

A release component is a component which was created by a specific, versioned release. For example, consider the 1.2.0 release of the commons-lang library stored in the Central Repository. This release component, commons-lang-1.2.0.jar, and the associated POM, commons-lang-1.2.0.pom, are static objects which will never change in the Central Repository. Released components are considered to be solid, stable and perpetual in order to guarantee that builds which depend upon them are repeatable over time. The released JAR component is associated with a PGP signature, an MD5, and a SHA checksum which can be used to verify both the authenticity and integrity of the binary software component.

Snapshot

Snapshot components are components generated during the development of a software project. A Snapshot component has both a version number such as 1.3.0 or 1.3 and a time-stamp in its name. For example, a snapshot component for commons-lang 1.3.0 might have the name commons-lang-1.3.0.-20090314.182342-1.jar the associated POM, MD5 and SHA hashes would also have a similar name. To facilitate collaboration during the development of software components, Maven and other clients that know how to consume snapshot components from a repository also know how to interrogate the metadata associated with a Snapshot component to retrieve the latest version of a Snapshot dependency from a repository.

A project under active development produces snapshot components that change over time. A release is comprised of components which will remain unchanged over time.

Looking at the Maven repository format and associated concepts and ideas allowed you grasp some of the details and intricacies involved with different tools and repository formats, that will help you appreciate the need for repository management.

Managing Repositories

The proliferation of different repository formats and tools accessing them as well as the emergence of more publicly available repositories has triggered the need to manage access and usage of these repositories and the components they contain.

In addition, hosting your own private repositories for internal components has proven to be a very efficient methodology to exchange components during all phases of the software development life cycle. It is considered a best practice at this stage.

The task of managing all the repositories your development teams interact with can be supported by the use of a dedicated server application - a repository manager.

Put simply, a repository manager provides two core features:

- the ability to proxy a remote repository and cache components saving both bandwidth and time required to retrieve a software component from a remote repository repeatedly
- the ability to host a repository providing an organization with a deployment target for internal software components

Just as Source Code Management (SCM) tools are designed to manage source code, repository managers have been designed to manage and track external dependencies and components generated by your build.

Repository managers are an essential part of any enterprise or open-source software development effort, and they enable greater collaboration between developers and wider distribution of software, by facilitating the exchange and usage of binary components.

Once you start to rely on repositories, you realize how easy it is to add a dependency on an open source software library available in a public repository, and you might start to wonder how you can provide a similar level of convenience for your own developers. When you install a repository manager, you are bringing the power of a repository like the Central Repository into your organization. You can use it to proxy the Central Repositories and other repositories, and host your own repositories for internal and external use.

Capabilities of a Repository Manager

In addition to these two core features, a repository manager can support the following use cases:

- allows you to manage binary software components through the software development life-cycle
- search and catalogue software components
- control component releases with rules and add automated notifications
- integrate with external security systems, such as LDAP or Atlassian Crowd
- manage component metadata
- host components, not available in external repositories
- control access to components and repositories
- display component dependencies

- browse component archive contents

Advantages of Using a Repository Manager

Using a repository manager provides a number of benefits including:

- improved software build performance due to faster component download off the local repository manager
- reduced bandwidth usage due to component caching
- higher predictability and scalability due to limited dependency on external repositories
- increased understanding of component usage due to centralized storage of all used components
- simplified developer configuration due to central access configuration to remote repositories and components on the repository manager
- unified method to provide components to consumers reducing complexity overheads
- improved collaboration due the simplified exchange of binary components

Software Supply Chain Automation

Once you start adopting a repository manager as a central point of storage and exchange for all components, expand its use in your efforts to automate and manage the software supply chain throughout your software development life-cycle.

Modern software development practices have shifted dramatically from large efforts of writing new code to the usage of components to assemble applications. This approach limits the amount of code authorship to the business-specific aspects of your software.

A large number of open source components in the form of libraries, reusable widgets, whole applications, or application servers are now available featuring very high levels of quality and feature sets that could not be implemented as a side effect of your business application development. For example, creating a new web application framework and business workflow system just to create a website with a publishing workflow would be extremely inefficient.

Development starts with the selection of suitable components for your projects based on comprehensive information about the components and their characteristics. Nexus Repository Manager Pro includes features, such as the display of security vulnerabilities as well as license analysis results within search results and the Repository Health Check reports for a proxy repository.

Software supply chain automation progresses through your daily development efforts, your continuous integration builds and your release processes all the way to your applications deployed in production environments at your clients or your own infrastructure.

Nexus IQ Server provides a number of tools to improve your component usage in your software supply chain allowing you to automate your processes to ensure high quality output, while increasing your development speed towards continuous deployment procedures. These include:

- integration with common development environments like the Eclipse IDE
- plugins for continuous integration servers such as Jenkins, Hudson or Eclipse
- visualizations in quality assurance tools like SonarQube
- command line tools for custom integrations
- notifications to monitor component flows

Nexus IQ Server enables you to ensure the integrity of the modern software supply chain, amplifying the benefits of modern development facilitating component usage, while reducing associated risks.

Installation

 Available in Nexus Repository OSS and Nexus Repository Pro

Installing and running Nexus Repository Manager is straight-forward. You can either unpack the archive in a directory to which you have full access, or you can install it with a Docker image.

Topics in the section:

- [Java Runtime Environment](#)(see page 169)
- [Installation Methods](#)(see page 170)
- [Run as a Service](#)(see page 172)
- [Run Behind a Reverse Proxy](#)(see page 176)
- [Accessing the User Interface](#)(see page 182)
- [Directories](#)(see page 183)
- [Configuring the Runtime Environment](#)(see page 185)
- [Post Install Checklist](#)(see page 190)
- [Securing Nexus Repository Manager](#)(see page 192)

Java Runtime Environment

Nexus Repository Manager requires a Java 8 Runtime Environment (JRE). The distributions for OSX and Windows include suitable runtime environments for the specific operating system. The distributions for Unix do not include the runtime environment. If you prefer to use an external runtime or use a Unix operating system, you can choose to install the full JDK or the JRE only. You can confirm the installed Java version with the `java -version` command, for example:

```
$ java -version
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

When multiple JDK or JRE versions are installed, you need to ensure the correct version is configured by running the above command as the operating system user that is used to run the repository manager.

In the event you have a non-standard location you need to update the configuration to specify a specific JDK or JRE installation path. To set the path for a specific Java location open the `bin/nexus` script and locate the line `INSTALL4J_JAVA_HOME_OVERRIDE`. Remove the hash and specify the location of your JDK/JRE:

```
INSTALL4J_JAVA_HOME_OVERRIDE=/usr/lib/jvm/openjdk-8
```

The startup script verifies the runtime environment by checking for the existence of the nested `bin/java` command as well as major and minor version of the runtime to be the required 1.8. If the configured runtime is not suitable, it will proceed with a best effort to locate a suitable runtime configured on the path or via the `JAVA_HOME` environment variable. If successful, it will start up the repository manager with this JVM. This allows you to have a dedicated runtime environment for the repository manager installed that is not on the path and not used by other installed applications. Further, you can separate upgrades of the Java runtime used by the repository manager from upgrades of the runtime used by other applications.

Installation Methods

Installing and Running with the Distribution Archive

The distribution archives combine the application and all required resources in an archive file.

If you are installing the repository manager on a local workstation to give it a test run, you can install it in your home directory or wherever you like. Nexus Repository Manager doesn't have any hard-coded directories and will run from any directory.

You can extract the archive ZIP for Windows with any archiving tool like [7zip](#)⁶⁴⁹ or on the command line.

On Windows you should install the repository manager outside Program Files to avoid problems with Windows file registry virtualization. If you plan to run the repository manager as a specific user you can install it into the AppData\Local directory of that users home directory. Otherwise simply use e.g., C:\nexus or something similar, ensuring that the user running the application has full access.

On OSX or Linux the downloaded GZip'd TAR archive can be extracted with the command `tar xvzf`.

⁶⁴⁹ <http://www.7-zip.org/download.html>

You install the repository manager in a directory other than your users home directory. On a Unix machine common practice is to use /opt. The extraction process creates two sibling directories: an application directory and a data directory, sometimes called the "Sonatype Work" directory. Further details about these folders and their contents can be found in [Directories\(see page 183\)](#).

The `bin` folder contains the generic startup scripts for Unix-like platforms called `nexus`. The Windows platform equivalent is called `nexus.exe`.

To start the repository manager from application directory in the `bin` folder on a Unix-like platform like Linux use:

```
./nexus run
```

The equivalent invocation on Windows requires a / in front of the `run` and any other commands.

```
nexus.exe /run
```

Starting the repository manager with the `run` command will leave it running in the current shell and display the log output.

Startup is complete when the log shows the message "Started Sonatype Nexus".

In order to shut down the repository manager running via the `run` command, you have to press CTRL-c. The `nexus` script can be used to manage the repository manager as a background application on OSX and Unix with the `start`, `stop`, `restart`, `force-reload` and `status` commands. The Windows `nexus.exe` command supports similar commands with a prefix of / e.g., `/start`. Once the repository manager is started you can access the user interface as detailed in [Accessing the User Interface\(see page 182\)](#).

- ⓘ To uninstall the repository manager from an archive installation, remove the service configuration and delete the entire directory.

Installing with Docker

Docker automates the deployment of applications inside virtualized Linux containers. You can create a container that supports the installation of Nexus Repository Manager Pro and Nexus Repository Manager OSS. To install the repository manager with a Docker image, follow the steps at [the Sonatype nexus3 Docker Hub image⁶⁵⁰](#).

⁶⁵⁰ <https://hub.docker.com/r/sonatype/nexus3/>

Run as a Service

When installing Nexus Repository Manager for production usage it has to be configured to run as a service, so it restarts after the server reboots. It is good practice to run that service or daemon as a specific user that has only the required access rights. Installation from the [distribution archive](#)(see page 22) does not include the configuration of a service. The following sections provide instructions for configuring the service manually. Independent of the operating system the steps are:

- Create operating system user with limited access rights dedicated to run the repository manager as a service
- Ensure suitable Java runtime environment is installed - see [Java Runtime Environment](#)(see page 169)
- Configure the service and ensure it starts as part of the operating system boot process

Linux

You can configure the repository manager to run as a service with `init.d` or `systemd`. Both are startup frameworks used in Linux-based systems such as Ubuntu and CentOS. They are, essentially, initscripts that load commands to manage the repository manager daemon. Before running the service configure an absolute path for your repository manager files. Then create a `nexus` user with sufficient access rights to run the service. In `bin/nexus.rc` assign the user between the quotes in the line below:

```
run_as_user="nexus"
```

Symlink \$installdir/bin/nexus to /etc/init.d/nexus:

```
sudo ln -s /opt/nexus-3.15.2-01/bin/nexus /etc/init.d/nexus
```

Adjust the location (shown above as "/opt/nexus-3.15.2-01") as needed for your installation location.

chkconfig

This example uses `chkconfig`, a tool that targets the initscripts in `init.d` to run the `nexus` service. Run these commands to activate the service:

```
cd /etc/init.d  
sudo chkconfig --add nexus  
sudo chkconfig --levels 345 nexus on  
sudo service nexus start
```

The second command adds nexus as a service to be started and stopped with the command. `chkconfig` manages the symbolic links in `/etc/rc[0-6].d` which control the services to be started and stopped when the operating system restarts or transitions between run-levels. The third command adds nexus to run-levels 3, 4 and 5. Then the service command starts the repository manager.

update-rc.d

This example uses `update-rc.d`, a tool similar to the `chkconfig`.

```
cd /etc/init.d  
sudo update-rc.d nexus defaults  
sudo service nexus start
```

In the second line you will run a default priority to add the `nexus` service before starting it.

systemd

This example is a script that uses `systemd` to run the repository manager service. Create a file called `nexus.service`. Add the following contents, then save the file in the `/etc/systemd/system/` directory:

```
[Unit]  
Description=nexus service  
After=network.target  
  
[Service]  
Type=forking  
LimitNOFILE=65536  
ExecStart=/opt/nexus-3.15.2-01/bin/nexus start  
ExecStop=/opt/nexus-3.15.2-01/bin/nexus stop  
User=nexus  
Restart=on-abort  
  
[Install]  
WantedBy=multi-user.target
```

Activate the service with the following commands:

```
sudo systemctl daemon-reload  
sudo systemctl enable nexus.service  
sudo systemctl start nexus.service
```

After starting the service for any Linux-based operating systems, verify that the service started successfully.

```
tail -f /opt/sonatype-work/nexus3/log/nexus.log
```

The tail command verifies that the service has been started successfully. If successful, you should see a message notifying you that it is listening for HTTP.

- ❗ Be sure to assign the appropriate permissions to the user running the nexus service.

PID File

When started as a service on Linux, NXRM will create a file which holds a process ID in the operating system /tmp directory. This file will have have a name of the form:

- prefix: "i4jdaemon_"
- suffix: the absolute path to the nexus start script with path part separators replaced with underscores
Example: Temporary directory is /tmp and path to start script is at /opt/nexus/nexus-3.14.0-04/bin/nexus , then the file created will be at "/tmp/i4jdaemon_opt_nexus_nexus-3.14.0-04_bin_nexus".

If the service pid file cannot be written the service startup will silently fail, without any logging statements written to the nexus.log.

If the NXRM process is already stopped, and the service is failing to start, first confirm there is no log output being added to the nexus.log file. If not, then check for an pre-existing pid file. If that file is present, delete it first before trying to start the service.

The directory this file is created in can optionally be changed by editing [\\$install-dir\(see page 183\)](#)/bin/nexus.vmoptions and adding a line like this:

```
-Dinstall4j.pidDir=/some/absolute/path/to/a/directory
```

The directory specified by the property must already exist for the property to work - the directory will not be created automatically.

Windows

The startup script that runs Nexus Repository Manager Pro and Nexus Repository Manager OSS on Windows platforms is bin/nexus.exe. The script includes standard commands for starting and stopping the service. It also contains commands install and uninstall to create and delete the configuration for the service. You can create the service configuration with:

```
nexus.exe /install <optional-service-name>
```

The new service is named `nexus` by default. It is available in the Windows console application to manage services such as Windows Services. You can start, stop and restart the service there as well as configure it to start as part of a operating system startup. Alternatively you can manage the service on the command line:

```
nexus.exe /start <optional-service-name>
nexus.exe /stop <optional-service-name>
nexus.exe /uninstall <optional-service-name>
```

The `<optional-service-name>` parameter with a value of e.g. `nexus3` can be used to create a service that does not collide with an existing service established for [Nexus Repository Manager 2](#)⁶⁵¹ running on the same server.

Mac OS X

The standard way to run a service on Mac OS X is to use `launchd`, a program that starts, stops and manages daemons and scripts in Apple OS X environments. To run the service you need to create an XML document called with the file extension `.plist` to define its properties. An example plist file for the repository manager installed in `/opt` is shown here:

A sample `com.sonatype.nexus.plist` file

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>com.sonatype.nexus</string>
  <key>ProgramArguments</key>
  <array>
    <string>/opt/nexus/bin/nexus</string>
    <string>start</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
</dict>
</plist>
```

After saving the file as `com.sonatype.nexus.plist` in `/Library/LaunchDaemons/` you have to change the ownership and access rights:

```
sudo chown root:wheel /Library/LaunchDaemons/com.sonatype.nexus.plist
sudo chmod 644 /Library/LaunchDaemons/com.sonatype.nexus.plist
```

⁶⁵¹ <https://help.sonatype.com/display/NXRM2>

Consider setting up a different user to run the repository manager and adapt permissions and the RUN_AS_USER setting in the nexus startup script. With this setup the repository manager starts as a service at boot time. To manually start it after the configuration you can use:

```
sudo launchctl load /Library/LaunchDaemons/com.sonatype.nexus.plist
```

Run Behind a Reverse Proxy

Nexus Repository Manager is a sophisticated server application with a web-application user interface, answering HTTP requests using the high-performance servlet container Eclipse Jetty. Organizations are sometimes required to run applications like Nexus Repository Manager behind a reverse proxy. Reasons may include:

- security and auditing concerns
- network administrator familiarity
- organizational policy
- disparate application consolidation
- virtual hosting
- exposing applications on restricted ports
- SSL termination
- SSO or SAML Integration

This section provides some general guidance on how to configure common reverse proxy servers to work with Nexus Repository Manager. Always consult your reverse proxy administrator to ensure your configuration is secure. The default webapp context path for the repository manager user interface is /. In the instance where the repository manager needs to be proxied at a different base path you must change the default path by editing a property value. In [Base URL Creation](#)(see page 263), follow the steps to change or update the base URL if you want an alternate server name.

In the following examples, review the sections on changing the HTTP port and context path to properly reverse-proxy the repository manager. Consult your reverse proxy product documentation for details: [Apache httpd](#)⁶⁵² ([mod_proxy](#)⁶⁵³, [mod_ssl](#)⁶⁵⁴), [nginx](#)⁶⁵⁵ ([ngx_http_proxy_module](#)⁶⁵⁶, [ssl compatibility](#)⁶⁵⁷).

⁶⁵² <http://httpd.apache.org/>

⁶⁵³ http://httpd.apache.org/docs/current/mod/mod_proxy.html

⁶⁵⁴ http://httpd.apache.org/docs/current/mod/mod_ssl.html

⁶⁵⁵ <http://nginx.org/en/docs/>

⁶⁵⁶ http://nginx.org/en/docs/http/ngx_http_proxy_module.html

⁶⁵⁷ http://nginx.org/en/docs/http/configuring_https_servers.html#compatibility

Example: Reverse Proxy on Restricted Ports

Scenario : You need to expose the repository manager on restricted port 80. The repository manager should not be run with the root user. Instead run your reverse proxy on the restricted port 80 and the repository manager on the default port 8081. End users will access the repository manager using the virtual host URL `http://www.example.com/` instead of `http://localhost:8081/`. Ensure your external hostname (`www.example.com`) routes to your reverse proxy server. This example uses the default content path (/).

Apache httpd

```
ProxyRequests Off
ProxyPreserveHost On

<VirtualHost *:80>
    ServerName www.example.com
    ServerAdmin admin@example.com

    AllowEncodedSlashes NoDecode

    ProxyPass / http://localhost:8081/ nocanon
    ProxyPassReverse / http://localhost:8081/
    ErrorLog logs/www.example.com/error.log
    CustomLog logs/www.example.com/access.log common
</VirtualHost>
```

nginx

```
http {  
  
    proxy_send_timeout 120;  
    proxy_read_timeout 300;  
    proxy_buffering off;  
    keepalive_timeout 5 5;  
    tcp_nodelay on;  
  
    server {  
        listen *:80;  
        server_name www.example.com;  
  
        # allow large uploads of files  
        client_max_body_size 1G;  
  
        # optimize downloading files larger than 1G  
        #proxy_max_temp_file_size 2G;  
  
        location / {  
            # Use IPv4 upstream address instead of DNS name to avoid attempts by nginx to use IPv6 DNS lookup  
            proxy_pass http://127.0.0.1:8081/;  
            proxy_set_header Host $host;  
            proxy_set_header X-Real-IP $remote_addr;  
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        }  
    }  
}
```

Example: Reverse Proxy Virtual Host at Custom Context Path

Scenario: You need to expose the repository manager using a custom host name `repo.example.com` on a restricted port at a base path of `/nexus`. Ensure your external hostname (`repo.example.com`) routes to your reverse proxy server and edit the webapp path a slash at end (/).

Apache httpd

```
ProxyRequests Off
ProxyPreserveHost On

<VirtualHost *:80>
    ServerName repo.example.com
    ServerAdmin admin@example.com

    AllowEncodedSlashes NoDecode

    ProxyPass /nexus http://localhost:8081/nexus nocanon
    ProxyPassReverse /nexus http://localhost:8081/nexus
    ErrorLog logs/repo.example.com/nexus/error.log
    CustomLog logs/repo.example.com/nexus/access.log common
</VirtualHost>
```

nginx

```
http {

    proxy_send_timeout 120;
    proxy_read_timeout 300;
    proxy_buffering off;
    keepalive_timeout 5 5;
    tcp_nodelay on;

    server {
        listen *:80;
        server_name repo.example.com;

        # allow large uploads of files
        client_max_body_size 1G;

        # optimize downloading files larger than 1G
        # proxy_max_temp_file_size 2G;

        location /nexus {
            # Use IPv4 upstream address instead of DNS name to avoid attempts by nginx to use IPv6 DNS lookup
            proxy_pass http://127.0.0.1:8081/nexus;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
    }
}
```

Example: Reverse Proxy SSL Termination at Base Path

Scenario: Your organization has standardized a reverse proxy to handle SSL certificates and termination. The reverse proxy virtual host will accept HTTPS requests on the standard port 443 and serve content from the repository manager running on the default non-restricted HTTP port 8081 transparently to end users. Ensure your external host name (repo.example.com) routes to your reverse proxy server and edit the webapp path to be slash (/). To test your configuration, review the steps to [generate a self-signed SSL certificate](#)⁶⁵⁸ for reverse proxy servers.

Apache httpd

The example assumes that Apache httpd has been [configured to load](#)⁶⁵⁹ `mod_ssl`⁶⁶⁰ and `mod_headers`⁶⁶¹ modules.

Example Apache httpd configuration for SSL termination at base Path

```
Listen 443

ProxyRequests off
ProxyPreserveHost On

<VirtualHost *:443>
    SSLEngine on

    SSLCertificateFile "example.pem"
    SSLCertificateKeyFile "example.key"

    AllowEncodedSlashes NoDecode

    ServerName repo.example.com
    ServerAdmin admin@example.com
    ProxyPass / http://localhost:8081/ nocanon
    ProxyPassReverse / http://localhost:8081/
    RequestHeader set X-Forwarded-Proto "https"

    ErrorLog logs/repo.example.com/nexus/error.log
    CustomLog logs/repo.example.com/nexus/access.log common
</VirtualHost>
```

⁶⁵⁸ <https://support.sonatype.com/hc/en-us/articles/213465768-SSL-Certificate-Guide>

⁶⁵⁹ <https://httpd.apache.org/docs/2.4/configuring.html#modules>

⁶⁶⁰ https://httpd.apache.org/docs/2.4/mod/mod_ssl.html

⁶⁶¹ https://httpd.apache.org/docs/2.4/mod/mod_headers.html

nginx

The example assumes that nginx has been compiled using the `--with-http_ssl_module`⁶⁶² option.

Example nginx configuration for SSL termination at base Path

```
http {  
  
    proxy_send_timeout 120;  
    proxy_read_timeout 300;  
    proxy_buffering off;  
    keepalive_timeout 5 5;  
    tcp_nodelay on;  
  
    server {  
        listen *:443;  
        server_name repo.example.com;  
  
        # allow large uploads of files  
        client_max_body_size 1G;  
  
        # optimize downloading files larger than 1G  
        #proxy_max_temp_file_size 2G;  
  
        ssl on;  
        ssl_certificate example.pem;  
        ssl_certificate_key example.key;  
  
        location / {  
            # Use IPv4 upstream address instead of DNS name to avoid attempts by nginx to use IPv6 DNS lookup  
            proxy_pass http://127.0.0.1:8081/;  
            proxy_set_header Host $host;  
            proxy_set_header X-Real-IP $remote_addr;  
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
            proxy_set_header X-Forwarded-Proto "https";  
        }  
    }  
}
```

Apache httpd with npm Repositories

Npm [scoped packages](#)⁶⁶³ use encoded slash characters ("/") in their URL's. By default Apache will not allow encoded slashes through. If you're using npm and have Apache running in front of Nexus add the following to your configuration so that it will allow encoded slashes through:

⁶⁶² <https://www.nginx.com/resources/wiki/start/topics/tutorials/installoptions/#optional-modules>

⁶⁶³ <https://docs.npmjs.com/misc/scope>

```
AllowEncodedSlashes NoDecode
```

The ProxyPass directive will also need nocanon option:

```
ProxyPass / http://localhost:8081/ nocanon
```

Note that the [documentation](#)⁶⁶⁴ for "noncanon" says "this keyword may affect the security of your backend". Sonatype server products do not rely on reverse proxies to filter out suspect URLs containing path info with encoded values, so it is OK to use "noncanon" with Nexus Repository Manager.

Apache httpd 2.0.52 to 2.2.8

If you are running Apache httpd 2.0.52 to 2.2.8 and you choose to set:

```
AllowEncodedSlashes On
```

Then this will trigger a bug that incorrectly decodes encoded slashes when they should not be:

https://bz.apache.org/bugzilla/show_bug.cgi?id=35256

Accessing the User Interface

Once the repository manager is started, the application is listening on the configured IP address range and port. By default any IP address and port 8081 are used. To access the web application user interface, fire up a web browser and type in the URL `http://<server_host>:<port>` e.g. `http://localhost:8081/`. If the repository manager started up successfully and network settings allow you to connect to the server, the user interface looks similar to *Figure: "Initial User Interface"*.

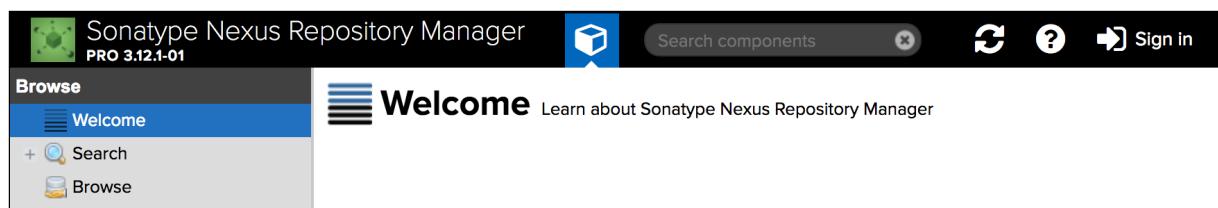


Figure: "Initial User Interface"

While the documentation uses `localhost` throughout, you may need to use the IP Loopback Address of `127.0.0.1`, the IP address or the DNS hostname assigned to the machine running the repository manager. The user interface shows the features available to an anonymous user. The repository manager installation

⁶⁶⁴ https://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxypass

includes an administrative user with full access. Its username is *admin* and the initial password can be found in an *admin.password* file in the *\$data-dir* directory. You can sign in with the button on the top right corner of the user interface.

Next steps after successfully accessing the user interface are detailed in [User Interface\(see page 209\)](#), [Configuration\(see page 233\)](#) and setting up whatever [Formats\(see page 364\)](#) you will use.

More information about security related topics can be found in [Security\(see page 309\)](#).

Directories

After you extract the repository manager archive, two directories will appear:

Installation directory

This directory contains the Nexus Repository Manager application and all the required additional components such as Java libraries and configuration files. The name of the directory by default uses *nexus-* and is appended with the version name. In this section, and throughout the documentation, it is referred to as *\$install-dir* in any code segments.

Data directory

This directory contains all the repositories, components and other data that are stored and managed by the repository manager. The default location of the data directory is *../sonatype-work/nexus3* relative to the installation directory. In this section, and throughout the documentation, it is referred to as *\$data-dir* in any code segments.

Installation Directory

The installation directory includes a number of nested directories:

```
$ ls -1 nexus-<version>
LICENSE.txt
NOTICE.txt
bin
deploy
etc
lib
public
system
```

LICENSE.txt and NOTICE.txt

These are files that contain legal details about the license and copyright notices

bin

This directory contains the **nexus** startup script itself as well as startup-related configuration files

etc

This directory contains configuration files

lib

This directory contains binary libraries related to Apache Karaf

public

This directory contains public resources of the application

system

This directory contains all components and plugins that constitute the application

Data Directory

Location

The data directory is found by default at `../sonatype-work/nexus3`.

To change from the default location see the [Configuring the Data Directory section of the Configuring the Runtime Environment\(see page 189\)](#) page.

Contents

The data directory includes subdirectories that contain all the components, repositories, configurations and other data persisted by the repository manager.

The subdirectories included are:

blobs/

The parent directory of all file system based blob stores that are not defined with an absolute storage path. For example, the default blob store will live at `../sonatype-work/nexus3/blobs/default`.

cache/

This directory contains information on currently cached Karaf bundles

db/

This directory contains the OrientDB databases which are the primary storage for your repository manager's metadata

elasticsearch/

This directory contains the currently configured state of Elasticsearch

etc/

This directory contains the main runtime configuration and customization of the repository manager. The files are explained further in [Configuring the Runtime Environment](#)(see page 185).

health-check/

This directory contains cached reports from the Repository Health Check feature

keystores/

This contains the automatically generated key used to identify your repository manager

log/

This directory contains several log files that capture information about various aspects of the running repository manager. The `nexus.log` and `request.log` files are rotated every day so this directory also contains archived copies of these files. To reclaim disk space, you can delete old log files from the logs directory. Log files found in this directory include:

- **`nexus.log`** - The main repository manager application log. Log messages contain standard log output fields including date/time, log level, the associated thread, class and message.
- **`request.log`** - Used to log http access requests to a running repository manager. Log messages contain information such as client host, user and HTTP request attributes including status code, bytes, and user-agent header.
- **`jvm.log`** - Contains JVM stdout, stderr and thread dump messages
- **`karaf.log`** - This is the Apache Karaf container log file which contains messages specific to the repository manager startup

The `log` directory also contains a **tasks** subdirectory which contains separate, uniquely named (by date, time and task name) log output files for each task that is run. See [Task Logging](#)(see page 279) for more details concerning naming strategy and content of these files.

tmp/

This directory is used for temporary storage

- ⚠** Unless you specify a relative path, running *Export configuration & metadata for backup* task creates a folder with a snapshot of the databases, in the data directory. See [Prepare a Backup](#)(see page 294) for the repository manager to learn how to configure this task.

Configuring the Runtime Environment

Configuring the specifics of the repository manager runtime involves configuration for all components in various configuration files and startup scripts. This section details these and provides recipes for specific tasks. The startup of the JVM running the repository manager is managed via files in the `$install-dir/bin` directory within the installation.

The application startup loads JVM arguments from the Java [properties](#)⁶⁶⁵ file `$install-dir/bin/nexus.vmoptions`:

Example `nexus.vmoptions` properties file content

```
-Xms2703m
-Xmx2703m
-XX:MaxDirectMemorySize=2703m
-XX:+UnlockDiagnosticVMOptions
-XX:+LogVMOutput
-XX:LogFile=../sonatype-work/nexus3/log/jvm.log
-XX:-OmitStackTraceInFastThrow
-Djava.net.preferIPv4Stack=true
-Dkaraf.home=.
-Dkaraf.base=.
-Dkaraf/etc/etc/karaf
-Djava.util.logging.config.file=etc/karaf/java.util.logging.properties
-Dkaraf.data=../sonatype-work/nexus3
-Dkaraf.log=../sonatype-work/nexus3/log
-Djava.io.tmpdir=../sonatype-work/nexus3/tmp
-Dkaraf.startLocalConsole=false
-Djava.endorsed.dir=lib/endorsed
```

The main location for configuration files is the `etc` directory. It includes one properties file and a number of nested directories:

```
$ ls -1 nexus-<version>/etc
fabric
jetty
karaf
logback
nexus-default.properties
ssl
```

fabric

Configuration files for Ehcache, Elasticsearch, and OrientDB

karaf

Configuration files for Apache Karaf, including:

- **config.properties** - The main configuration for the Apache Karaf runtime. This file should not be modified.
- **custom.properties** - Customizable configuration used by Apache Karaf. This file can be used to pass additional parameters to the Apache Karaf container.

⁶⁶⁵ <https://docs.oracle.com/javase/8/docs/api/java/util/Properties.html#load-java.io.Reader->

- **org.apache.*** and **org.ops4j.*** - various Karaf and OSGi related configuration files
- **system.properties** - system properties used for the JVM and application start up

jetty

Configuration files for Eclipse Jetty

logback

Configuration files for logback including:

- **logback.xml** - Contains default logger definitions for log file names, log levels, pattern layouts and rotation rules for the `nexus.log` and `tasks/*.log` files
- **logback-access.xml** - Contains the default logger definition for the log file name, log level, pattern layout and rotation rules for the `request.log` file

See [Logging and Log Viewer](#)⁶⁶⁶ for more information on repository manager logging and the recommended approach for adjusting the logging configuration to meet your needs.

nexus-default.properties

This file should not be modified. A sample of default property values are listed in this file such as the main HTTP connector port and the webapp context path.

Change or add new configuration properties in `$data-dir/etc/nexus.properties`.

The file `$data-dir/etc/nexus.properties` is created the first time NXRM is started. If configuration is needed before first start then manually create `$data-dir/etc/nexus.properties` and add customizations there.

ssl

A directory to put keystores when configuring HTTPS

Configuring Memory

Before changing the default memory settings, [review our memory guidelines](#)(see page 142). Changing any one setting will require adjusting all 3 settings. The `-Xms` and `-Xmx` settings must always have the same value.

To change JVM memory

1. Open `$install-dir/bin/nexus.vmoptions` in a text editor
2. Find the lines which configure memory:

⁶⁶⁶ <https://help.sonatype.com/display/NXRM3M/Configuration#Configuration-LoggingandLogViewer>

Default memory settings

```
-Xms2703M  
-Xmx2703M  
-XX:MaxDirectMemorySize=2703M
```

3. Edit the values of each setting using the following format. The `size` is a number and the literal unit letters represent gigabytes (`g` , `G`) , megabytes (`m` , `M`), or kilobytes (`k` , `K`).
`-Xmssize[g|G|m|M|k|K]`
`-Xmxsize[g|G|m|M|k|K]`
`-XX:MaxDirectMemorySize=size[g|G|m|M|k|K]`
4. Save the file with permission and ownership of the user that will own the repository manager process.

Configuring Java Tool Options

The file `$install-dir/bin/nexus.vmoptions` specifies [java tool arguments](#)⁶⁶⁷, one per line, in [Java properties file format](#)⁶⁶⁸.

To edit java options

1. Open `$install-dir/bin/nexus.vmoptions` in a text editor
2. Specify each java option on a new line.
Make sure the last property is followed by an end of line character or a new blank line. Otherwise the last specified property will not be loaded.
3. Save the file with permission and ownership of the user that will own the repository manager process.

Changing the HTTP Port

The default value for the HTTP port used to access the repository manager user interface and resources is 8081. Therefore the user interface would be available at `http://localhost:8081/`. To change or update the port, locate the line `application-port=8081` in `$data-dir/etc/nexus.properties`, then edit the number. Here is an example where you would change the port to 9081:

```
application-port=9081
```

Therefore, the exposed URL will be `http://localhost:9081/`.

⁶⁶⁷ <https://docs.oracle.com/javase/8/docs/technotes/tools/unix/java.html#BGBCIEFC>

⁶⁶⁸ <https://docs.oracle.com/javase/8/docs/api/java/util/Properties.html#load-java.io.Reader->

Or setup HTTPS

Additionally, or instead, your use case may require you to run your server on HTTPS. This includes both business policy and usage of some NXRM features or repository formats. See [our security article on setting up SSL\(see page 338\)](#) for the means to do this.

Changing the Context Path

To change or update the context path in the instance you want point to a specific webapp or component, locate the `nexus-context-path=/` line in the `$data-dir/etc/nexus.properties`. Here is an example where you expose the user interface to a `components` directory.

```
nexus-context-path=/components/
```

Therefore, if the port is set to 9081, the exposed URL will be `http://localhost:9081/components/`.

Configuring the Data Directory

You can use `$install-dir/bin/nexus.vmoptions` to define a new location for data you want to preserve. In the configuration file change the values of `-Dkaraf.data`, `-Djava.io.tmpdir`, `-Dkaraf.log` and `-XX:LogFile` to designate an absolute path you prefer to use. The nexus service will look to add the data directory to the absolute path that you configure. For example, to use the absolute path `/opt/sonatype-work/nexus3` change the values as follows:

```
-Dkaraf.data=/opt/sonatype-work/nexus3  
-Djava.io.tmpdir=/opt/sonatype-work/nexus3/tmp  
-XX:LogFile=/opt/sonatype-work/nexus3/log/jvm.log  
-Dkaraf.log=/opt/sonatype-work/nexus3/log
```

Configuring the Temporary Directory

Repository manager uses the Java virtual machine temporary directory for important runtime files.

The temporary directory is set to be inside of the Nexus data directory at `$data-dir/tmp`. The advantage is file ownership and permissions can be isolated under a single directory tree.

If you wish to change the default temporary location, keep in mind the following requirements:

- the tmp directory requires exec permissions by the user owning the repository manager process. Do not use the noexec mount option on Linux for the location that contains the tmp directory. If noexec is set on the tmp directory, repository manager startup will fail with an `java.lang.UnsatisfiedLinkError` message of failed to map segment from shared object:

Operation not permitted . If the repository manager docker image is being used, one may need to adjust the permissions on the host directory that's being mounted as a docker volume to allow execution.

- do not manipulate any files put into the temp directory, especially when repository manager is running
- make sure the disk hosting the directory has at least 1GB free space

To Change the Temporary Directory

1. Open `$install-dir/bin/nexus.vmoptions` in a text editor
2. Edit the line which contains `-Djava.io.tmpdir=.../sonatype-work/nexus3/tmp` . The value can either be an absolute path or a path relative to the `$install-dir/bin` directory.
3. Save the file with permission and ownership of the user that will own the repository manager process.

Post Install Checklist

Nexus Repository Manager Pro and Nexus Repository Manager OSS ship with some default settings that need to be changed for your installation to be complete and secure. After installing and running the repository manager, you need to make sure that you complete the following tasks:

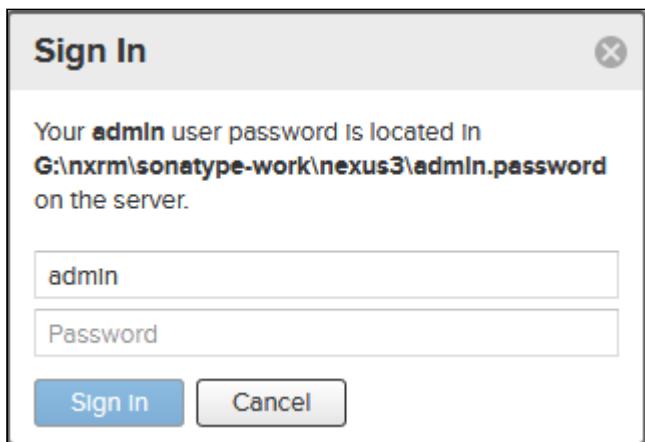
Step 1: Change the admin password

To ensure the system begins with a secure state, Nexus Repository Manager generates a unique random password during the system's initial startup which it writes to the data directory (`$data-dir`, typically `sonatype-work/nexus3`) in a file called `admin.password`.

To configure the system, the user must prove they're the system owner by retrieving the initial password from the filesystem. This password can then be changed manually through the user interface, or programmatically through the REST API.

Using the User Interface

Click the *Sign in* button in the top right of the page, until the initial password has been changed the dialog that comes up will indicate the file path of a file containing the password for the `admin` user.



The *admin* user will then be presented with an setup wizard which will assist the user in changing the admin password along with other initial system setup.

Using the REST API

For systems that are provisioned programmatically we also provide a REST API which can be used to change a user's password. The provisioning system must first read the randomly generated password from the `admin.password` file.

Step 2: Configure Anonymous Access

When using the user interface another step in the wizard is configuring whether [anonymous access](#)(see page 323) is allowed in the system. Until configured the system will allow unauthenticated users to read the contents of repositories.

Step 3: Change the Administrative Email Address

The *admin* user comes with a default email address configured and, since this address is not going to be very useful to anyone, see [Working with Your User Profile](#)(see page 229) for details.

Step 4: Configure the SMTP Settings

The repository manager can send username and password recovery emails. To enable this feature, you will need to configure a SMTP Host and Port as well as any necessary authentication parameters that the repository manager needs to connect to the mail server.

Step 5: Configure Default HTTP and HTTPS Proxy Settings

In many deployments the internet, and therefore any remote repositories that the repository manager needs to proxy, can only be reached via a HTTP or HTTPS proxy server internal to the deployment company. In these cases the connection details to that proxy server need to be configured in order for the repository manager to be able to proxy remote repositories at all.

Step 6: Setup a Backup procedure for your server

Read and utilize [Backup and Restore](#)(see page 294). Things happen and it is always advisable to backup your configurations and data on a scheduled basis.

Securing Nexus Repository Manager

Below are a few suggestions for making your Nexus Repository Manager (NXRM) instance more secure.

Limit IPs that can be reached from your NXRM host

NXRM can be configured by an administrator to contact internal and external IPs for various reasons such as retrieving certificates, creating proxy repositories, dispatching events to remote URLs and so on. You may limit the IPs that can be reached from the host machine running your NXRM instance but note that doing so could block the main use case for some features. For example, webhooks give administrators a way of integrating NXRM with other systems (e.g an auditing system, another NXRM instance, or a lightweight listener potentially on the same host), typically in the same data center. Hence, limiting webhook destinations to, for example, IPs external to your data center effectively blocks the main use case for them.

Privileges and service account

- Only assign the least necessary privileges to NXRM users.
- Create a dedicated operating system service account for running NXRM - do not run as the root user.
In addition, the service account must have read/write permissions to the \$install-dir and sonatype-work directories and must be able to create a valid shell. Please see [here](#)(see page 138) for detailed operating system service account recommendations.

Containerization

Running NXRM in a Docker container may reduce the impact of a successful attack. Without containerisation, if a malicious person successfully exploits a service and gains root access, they could do damage to other services running on the host. On the other hand, containerising means a successful attack on

that service is restricted to the container running that service. The NXRM3 docker images can be found at: <https://hub.docker.com/r/sonatype/nexus3/>.

Upgrading

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

Upgrading Nexus Repository Manager presents a necessary step to gain access to new features, bug fixes, performance improvements and other advantages. Regular updates to the latest release are recommended as a general best practice.

Specifically Nexus Repository Manager 3 represents a shift in design that supports a wider set of features requested by customers as well as a platform for a modern, expanded set of functionality. Given these changes, many to the core architecture, the upgrade process requires more attention than in previous versions.

This section covers upgrades of Nexus Repository Manager in general with a focus on upgrading Nexus Repository Manager version 2 to Nexus Repository Manager version 3. The general process of upgrading depends on the specific usage of the repository manager, its configuration and integration with other tools and is potentially complex. Further resources can be found in the [Support Knowledge Base](#)⁶⁶⁹.

-  Nexus Repository Manager Pro customers can take advantage of the assistance of the support team.

Why Upgrade to Nexus Repository Manager 3?

In Nexus Repository Manager 3, there is wider feature and functionality equivalency to Nexus Repository Manager 2. Highlights of new functionality include:

- Expanded repository format support
- Improved user interface
- Powerful component search
- Universal repository browsing
- Enhanced metadata

Of course, the choice to upgrade depends on the features your team is using and planning to use. In many cases upgrading to version 3 provides an enhanced set of features to support modern development

⁶⁶⁹ <https://support.sonatype.com/hc/en-us/sections/204911768>

practices and automation. However, it is a good idea to review the support site to compare [supported formats](#)(see page 197), [version 2 to 3 feature equivalency](#)(see page 148) and the [feature matrix](#)(see page 153).

Important

Upgrading Nexus Repository Manager 2 to 3 only provides native tooling to transfer content and configurations from the respective source repository manager to the target repository manager. We strongly discourage you to run the upgrade from version 2 to version 3 while simultaneously running any data center-to-data center transfers (e.g. synchronizing applications in your cloud server to on-premises data centers, or vice-versa).

Upgrading from 3.x to 3.y

Version 3 upgrades are supported from version **3.0.0 milestone 7 and later**.

Upgrading a Standalone Instance

Refer to our detailed [knowledge base article](#)⁶⁷⁰.

Upgrading a High Availability Cluster (HA-C)

Refer to specific instructions for [upgrading your HA-C environment](#)(see page 563).

Upgrading from 2.x to 2.y

At a high level, upgrading from a 2.x release of Nexus Repository Manager to a newer 2.y version typically includes:

- Extracting the new release bundle
- Replicating configuration changes
- Stopping 2.x instance
- Replacing the application directory with the new instance
- Starting the new instance

Full instructions are available on the [support site](#)⁶⁷¹.

⁶⁷⁰ <https://support.sonatype.com/hc/en-us/articles/115000350007>

⁶⁷¹ <https://support.sonatype.com/hc/en-us/articles/213464138>

Upgrading from 2.x to 3.y

- ✓ Upgrade should be done using only the latest 2.x and 3.y to assure that any fixes in the upgrade process are utilized when upgrading. The latest Nexus Repository Manager versions are verified as compatible before deployment. The first step in upgrading from 2.x to 3.x needs to be upgrading to the latest 2.x version. Only after this has been done should upgrade be attempted.

Upgrading from Nexus Repository Manager 2 to 3 requires you to be on the same licensed model of the product on both servers. Thus Nexus Repository Manager 2 OSS users should upgrade to Nexus Repository Manager 3 OSS and Nexus Repository Manager 2 PRO users should upgrade to Nexus Repository Manager 3 PRO. The same license file can be used for both instances. PRO users who have questions about their licensing can ask their Customer Success representative or file a support ticket.

Upgrading from Nexus Repository Manager 2 to 3 also requires the involved repository managers to use a compatible version of Nexus Repository Manager 2 and 3. If the source repository manager uses a version prior to 2.14.1, you must upgrade it as detailed in [Upgrading from 2.x to 2.y\(see page 194\)](#) before starting the upgrade to Nexus Repository Manager 3. The target repository manager is typically a fresh installation with a minimum release version of 3.1. If an existing Nexus Repository Manager 3 is used as the target, it has to be upgraded to 3.1 (or later) as documented in [Upgrading from 3.x to 3.y\(see page 194\)](#).

- ❗ Using an existing installation of Nexus Repository Manager 3 populated with data and configuration as the target repository manager is not supported. The upgrade should be done against a new Nexus Repository Manager 3 installation (a single node, not HA) that does not contain existing configuration. **Note that this means the upgrade must be done in one shot, running the upgrade multiple times is not supported.**

If you must upgrade using an older version of Nexus Repository Manager 2, against the recommendations of Sonatype, see the [compatibility matrix\(see page 146\)](#) to make sure you upgrade to the correct associated version. Upgrading using non-associated versions will result in errors. Also remember, only versions 2.14.1 and beyond can be upgraded to Nexus Repository Manager 3.

The following topics in this section go into Upgrading from 2.x to 3.y in more detail:

- [Upgrade Process and Expectations\(see page 196\)](#)
- [Data Transfer Methods\(see page 199\)](#)
- [Upgrade Details for Specific Elements\(see page 200\)](#)
- [Security Compatibility\(see page 202\)](#)

- Optimization, Performance, and Tuning(see page 202)
- Upgrade Procedures(see page 203)

Upgrade Process and Expectations

The process of upgrading Nexus Repository Manager 2 to 3, is similar to any major enterprise application upgrade, and should be managed via an upgrade plan. The upgrade plan is really just a specific checklist of all the steps required to perform the upgrade.

While the upgrade process is underway, you can continue to use the source Nexus Repository Manager 2. Any repository content that's added, updated, or deleted is picked up by the upgrade and added to the target Nexus Repository Manager 3 – however, configuration changes are not. Thus, you should not make changes to items such as realm settings, permissions, roles, role assignments, HTTP configuration, SSL certificates, or add new repositories. These types of configuration changes are not taken into account for an ongoing upgrade and can cause the upgrade process to fail.

What Is Upgraded

Nexus Repository Manager 3 represents an application design shift, involving a new architecture that supports advanced features for today's development practices. As such, a number of core changes to data stored occur as part of the upgrade process.

This includes:

Component storage format from files to blobs

Components in Nexus Repository Manager 2 are stored as individual files on disk. Version 3 stores components as blobs. The conversion process requires version 3 to iterate over every component stored in version 2. This takes the bulk of the time required for the upgrade process.

Settings and metadata

Settings and some component metadata in Nexus Repository Manager 2 are stored across many files. Conversely, Nexus Repository Manager 3 loads equivalent data into an OrientDB database.

URL

By default, Nexus Repository Manager 2 uses a different URL pattern to expose repositories and repository groups than Nexus Repository Manager 3. A capability is added during upgrade to allow your automated tools and CI to use the old patterns.

What Is Not Upgraded

The file structures within your repository manager environment differs between version 2 and version 3. Before preparing for the upgrade process, review this list of settings and configuration items. These items are not automatically included when you upgrade:

- custom branding
- virtual repositories
- Java VM settings, including custom system properties or variables
- operating system nexus service scripts
- operating system optimization, such as increasing allowable open file handles
- environment variables affecting values used to control the repository manager
- third-party or custom-developed plugins
- Jetty server XML configuration files
- repository formats unimplemented in NXRM3
- manual edits to other files under the nexus installation directory, such as edits to `nexus/WEB-INF/classes/ehcache.xml`
- custom log levels or edits to `logback.xml` configuration files (e.g. custom log file rotation, file naming, log patterns)
- tasks (called scheduled tasks in NXRM2)
- capabilities
- the `admin` user password from NXRM2
- staging repositories, settings, or configuration
- some existing artifact information based on NXRM2 data such as `uploadedBy`, `uploadedDate`, `lastModified` (is replaced by "system" value and date of migration)

There are equivalent configurations possible for all these values and customization in Nexus Repository Manager 3. The specifics vary widely and have to be applied manually after determining the need for such customization and developing specific plans for the modifications. The scope of these modifications varies from zero to large efforts. E.g. some VM start-up parameters might not be appropriate any more due to optimized performance of version 3. On the other hand custom plugin features might now be a standard supported features or require a completely new development effort.

Repository Format Support

Nexus Repository Manager 3 supports an evolving set of [repository formats](#)(see page 364). The automatic repository upgrade procedure from 2 to 3 has certain limitations.

Upgradable Formats

- Maven2 (excluding staging repositories)

- npm
- NuGet
- Site (called Raw in NXRM3)
- RubyGems

Upgrade with Limited Support

- YUM

RPMs stored in YUM enabled Maven 2 format repositories can be migrated to Maven2 format NXRM3 repositories only. Any YUM related capabilities must be disabled on the Maven 2 repository first.

There is no automatic option to [migrate RPMs in NXRM2 Maven2 repositories into YUM format repositories in NXRM 3](#)⁶⁷².

Upgrade Not Supported

- Maven1
- P2
- OBR

Designing Your Upgrade Plan

When upgrading, the most critical decisions you need to make about an upgrade plan are:

- Identification of a maintenance window for version 2, allowing the upgrade to proceed without interruption.
- Selection of an installation scenario that best supports your upgrade plan.
- Selection of an upgrade method.
- Getting access to a system storage , as well as location for content to be transferred to.
- Identification of configurations that may result in failure, and prevent upgrade of certain components.
- Review of security settings , and associated differences between version 2 and version 3.
- Considerations for optimization.

Supported Installation Scenarios

There are two scenarios for upgrading:

- Separate servers for version 2 and version 3
- Version 2 and version 3 running on the same server

⁶⁷² <https://issues.sonatype.org/browse/NEXUS-16670>

Data Transfer Methods

Upgrade is made possible by specific capabilities in version 2 and version 3 called *Upgrade: Agent* and *Upgrade* respectively. These capabilities manage the communication between the two servers and can transfer all configuration via web protocols. The bulk of the data to be transferred consists of all the binary components in the repositories that are upgraded. Once the *Upgrade: Agent* and *Upgrade* capabilities, mentioned in [Starting the Upgrade\(see page 204\)](#), are enabled and both repository manager instances are communicating, you can choose one of three methods for this transfer:

- [HTTP Download\(see page 199\)](#)
- [File System Copying\(see page 199\)](#)
- [File System Hard Linking\(see page 199\)](#)

HTTP Downloading

 When running the HTTP download method, we discourage you from synchronizing repository manager content to cloud services or on-premises data centers. This tool is solely designed to allow for data and configuration transfers between Nexus Repository Manager 2 to 3.

HTTP download is a transfer method in which version 3 makes HTTP requests to version 2 to transfer configuration and data. This is the slowest option.

If version 2 and version 3 are running on different machines and do not share access to the same file system storage, you must use the HTTP download method.

File System Copying

In this transfer method, version 2 tells version 3 the path of the file content to transfer and a simple file system copying is performed.

This upgrade method works if versions 2 and version 3 are configured to access the same storage system on identically named mount points. It is a faster process than the HTTP Download method, and has less impact on the performance of version 2.

File System Hard Linking

In this transfer method, version 2 tells version 3 the path of the file content to transfer and a file system hard link to the same content is created.

This upgrade method works if versions 2 and version 3 are configured to access the same storage system on identically named mount points and hard linking is supported by the file system used. It is the fastest transfer method.

File System Considerations

While discussed in greater detail in [Blob Stores\(see page 235\)](#), Nexus Repository Manager 3 allows you to create and name multiple blob stores to store your content. Before you start the upgrade process it is important to consider how you want to allocate space within the storage system.

When upgrading, make sure you have enough storage capacity in the destination file system(s). For instance, if you are using hard linking, the data is not duplicated. This saves storage space, but you must ensure that there are enough file handlers available for the content you want to transfer during the upgrade process. On the other hand, file copy and downloading do duplicate your data so upwards of double the original storage space can be, at least temporarily, needed.

Upgrade Details for Specific Elements

Due to fundamental changes in file structure between Nexus Repository Manager 2 and 3, you should review and compare the configuration details to prevent any failures.

Repository IDs

The Repository ID defined in version 2 is used as the Name for the upgraded repository in Nexus Repository Manager 3 as they define the access URL in both cases. The user-facing Repository Name from version 2 is dropped in the upgrade.

In addition note that IDs of repositories and repository groups in version 2 that differ only by case will not be accepted during an upgrade to version 3 (example version 2 IDs: myrepoid vs Myrepoid). To resolve the ID conflict review and change any IDs in version 2 to distinguishable names.

Repository Groups

Review the contents of the repository groups defined in Nexus Repository Manager 2 to ensure its contents are selected for upgrade. A single repository within the group, not selected, may prevent the entire group from being upgraded.

User Tokens

The upgrade tool only upgrades pre-existing user tokens to version 3, if user token support is enabled in version 2. In version 2, click the *User Token* tab, in the *Administration* menu, and enable the setting if you desire upgrade of user tokens.

Repository Health Check and SSL Health Check

You can include both your existing Repository Health Check and its corresponding SSL trust store configuration when you upgrade. If you are a Nexus Repository Manager OSS user you only have the ability to upgrade your settings from the *Health Check: Configuration* capability.

If you are a Nexus Repository Manager Pro user, you can also upgrade your existing *SSL: Health Check* settings. After the upgrade process is complete, settings for both *Health Check: Configuration* and *SSL: Health Check* capabilities are enabled in version 3, as they were in version 2.

NuGet API Key

The upgrade tool adds all keys to version 3 that are present in version 2 when asked, even if the NuGet API Key Realm is not active in version 2.

HTTP(S) Proxy Configuration

In general, your HTTP or HTTPS proxy settings for Nexus Repository Manager 2 may not be valid for your Nexus Repository Manager 3 environment. So you need to configure your HTTP or HTTPS proxy settings in version 3 in order to upgrade them to version 2.

If HTTP or HTTPS proxy settings were enabled in your source Nexus Repository Manager 2, the upgrade to your target Nexus Repository Manager 3 might fail because the target could not communicate with the source repository manager. That's because version 3 could not find a version 2 proxy server in place. Therefore if the HTTP or HTTPS settings were enabled in version 2 during an upgrade, version 3 would use its original HTTP or HTTPS settings, ignoring the settings in place for version 2. Additionally, a warning would be generated in the log if that error occurred.

Licensing

Any Nexus Repository Manager license you have will be applicable between version 2 and version 3. If you are upgrading your versions on the same server, no additional work need be done. If you are upgrading your Nexus Repository Manager 3 to a different server than Nexus Repository Manager 2, then you must reinstall the license on the new server, however, the license files are the same. You do not need a different license for Nexus Repository Manager 3 use.

IQ Server

If you are a Nexus Repository Manager Pro user, and you want to upgrade your source Nexus IQ Server settings and configuration to your target repository manager, ensure that your licenses include the integration for both versions. Your configuration for *IQ Server URL*, *Username*, *Password*, and *Request Timeout* will be included in the upgrade. Additional configuration, such as analysis properties, trust store usage, and the enabled Nexus IQ Server connection itself will be upgraded from versions 2 to 3.

Repositories that have been configured with the *IQ: Audit and Quarantine* capability will keep the capabilities as well as the audited and quarantined items during the upgrade.

- ❗ While Nexus Repository Manager 2 can be kept running while the upgrade occurs, changes to the content will only be moved up through the *Synchronization* upgrade step. Changes to audited and quarantined items from the IQ Server are included in this caveat.

Staging

NXRM 3 staging has been completely redesigned and is not compatible with NXRM 2 staging. Upgrading from NXRM 2 to NXRM 3 staging is not supported. Please see the [Staging\(see page 352\)](#) documentation for more details.

Security Compatibility

Before you upgrade from version 2 to version 3 review the differences in security settings along the upgrade path. Known changes may affect privileges, roles and repository targets.

Version 2 Roles

Roles upgraded from version 2 are assigned a Role ID that starts with nx2- in Nexus Repository Manager 3. Role descriptions created during the upgrade process have the word legacy in their description.

Version 2 Repository Targets and Target Privileges

Repository targets from version 2 are converted to [Content Selectors\(see page 251\)](#) in version 3. Repository targets were based on regular expressions, and are automatically converted to CSEL (Content Selector Expression Language), a performant subset of JEXL.

Optimization, Performance, and Tuning

When considering upgrade time and speed, take into account all enabled features on your Nexus Repository Manager 2 instance that you may not need. You can optimize the performance of your upgrade by disabling

specific features. As discussed in this article about [performance and tuning](#)⁶⁷³, identify and then reduce your list of used features to improve the performance of your repository manager. See some highlights, below:

System feeds

If your organization does not rely on system feeds, often used for team communication, learn how to disable them.

Repair index tasks

These tasks support searching components within the user interface, and do not need to be rebuilt that often, consider disabling them across all repositories.

Snapshot removal tasks

Enable both *Remove Snapshots from Repository* and *Remove Unused Snapshots From Repository*, which deletes old component versions no longer needed.

Repositories no longer supported

Remove any deprecated repositories. For example, any Maven 2 proxy repositories with the domain name "[codehaus.org](#)" should be deleted⁶⁷⁴.

Rebuild Maven Metadata Files

This scheduled task should only be run if you need to repair a corrupted Maven repository storage on disk.

Staging rules

If you are a Nexus Repository Manager Pro user that uses the application for staging releases, redefine or reduce the number of configured staging rules and staging profiles.

Scheduled tasks for releases

If you find empty *Use Index* checkboxes under *Task Settings*, use the opportunity to disable or remove those specific tasks for releases.

To help you decide how to reduce scheduled tasks, improving the performance of your upgrade, review the [knowledge base article](#)⁶⁷⁵.

Upgrade Procedures

⁶⁷³ <https://support.sonatype.com/hc/en-us/articles/213465138>

⁶⁷⁴ <https://support.sonatype.com/hc/en-us/articles/217611787>

⁶⁷⁵ <https://support.sonatype.com/hc/en-us/articles/213465208>

Starting the Upgrade

After you've designed your upgrade plan, considered system performance, and assessed storage needs, there are a few basic steps to start the upgrade:

1. Upgrade your existing version 2 instance to the latest available 2.x version as documented in [Upgrading from 2.x to 2.y \(see page 194\)](#).
2. If you have not already, install Nexus Repository Manager 3.
3. Enable the upgrade capabilities in both version 2 and version 3.

With the above complete, you can use the upgrade tool in version 3, which guides you through upgrading in three phases:

- *Preparing*, the phase that prepares the transfer and creation of all configuration and components.
- *Synchronizing*, the phase that counts and processes all components set to upgrade and upgrades all other configuration.
- *Finishing*, the phase that performs final clean up, then closes the process.

Enabling the Upgrade Capability in Version 2.x

In version 2, enable the *Upgrade: Agent* capability to open the connection for the *upgrade-agent*.

Follow these steps:

1. Click to expand *Administration* in the left-hand panel.
2. Click the *Capabilities* menu item to open the respective screen.
3. Click the *New* button to access the Create new capability modal.
4. Select *Upgrade: Agent* as your capability *Type*.
5. Click *Add* to close the modal and add the capability.
6. Copy and save the *Access Token* found on the *Status* tab for your new capability. You need it to configure the Upgrade tool in version 3.

In the lower section of the *Capabilities* interface, the repository manager acknowledges the upgrade-agent as *Active*.

Enabling the Upgrade Capability in Version 3.x

In version 3, enable the *Upgrade* capability to open the connection for the upgrade-agent and access the Upgrade tool.

Follow these steps:

1. Click *Capabilities* in the *System* section of the *Administration* main menu to open the Capabilities feature view.

2. Click *Create capability*.
3. Select *Upgrade*, then click *Create capability* to enable the upgrade capability.

Upgrading Content

After you enable the upgrade capabilities, access the upgrade tool in Nexus Repository Manager 3 to start your upgrade.

1. Go to the *Administration* menu.
2. Select *Upgrade* located in the *System* section of the *Administration* main menu to open the wizard.

Overview

The upgrade tool provides an overview of what is allowed for an upgrade as well as warnings on what cannot be upgraded.

Agent Connection

This screen presents three fields: *URL*, *Access Token*, and *Fetch Size*. In the *URL* field, enter the URL of your version 2 server including the context path e.g. `http://localhost:8081/nexus/`. In the *Access Token* field, enter the access token, copied from your version 2 *Upgrade: Agent capability Status* tab. The *Fetch Size* determines how many assets are transferred at once from version 2 server to version 3; lowering the default value can mitigate memory or time out issues.

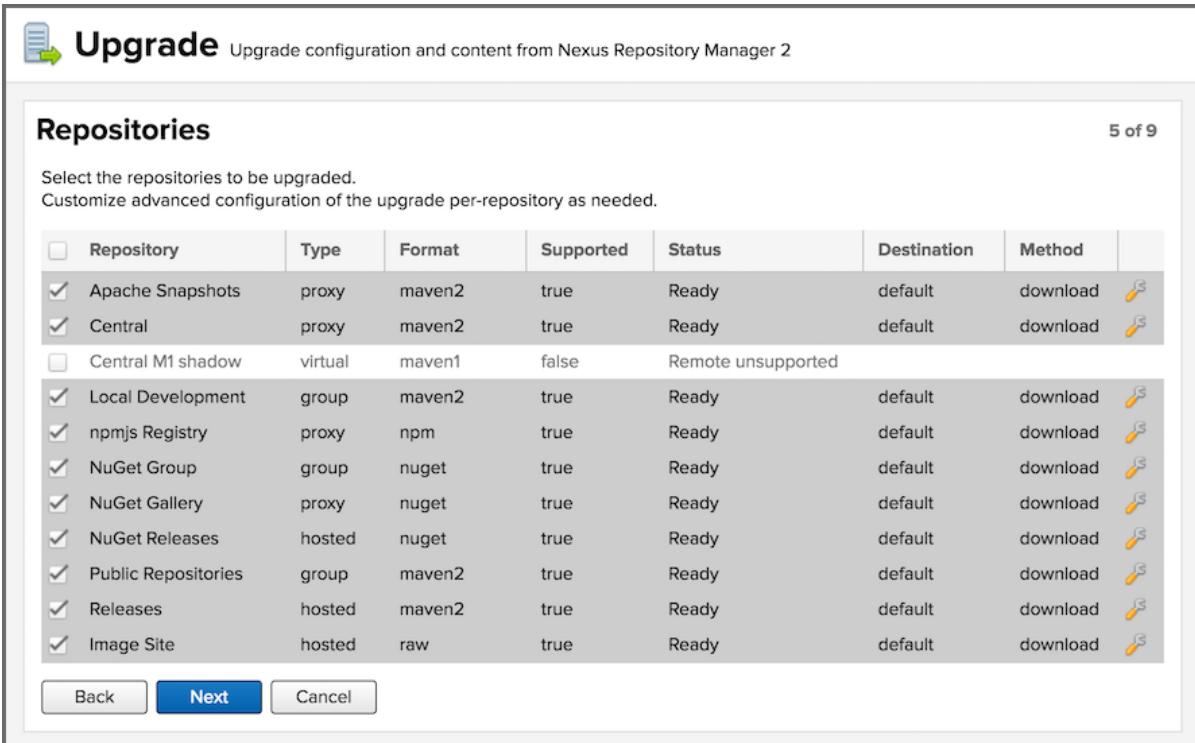
Content

This screen allows you to select from *Repository configuration and content*, which includes user accounts and associated security settings, and *Server configuration*. Check the options that apply.

- ⓘ The Repository configuration and content upgrades all user accounts when selected. If certain user accounts aren't desired in Nexus Repository Manager 3 then you can delete them from Nexus Repository Manager 3 after the upgrade for all repositories is done.

Repository Defaults

The Repository Defaults screen allows you to select the directory destination and transfer method. The first drop-down menu, *Destination*, allows you to pick a blob store name different from default. The second drop-down menu, *Method*, allows you to choose the transfer method. This section allows you to click and change each repository's individual transfer method and destination (i.e. blob store).



The screenshot shows the 'Upgrade' configuration screen for Nexus Repository Manager 3. The title bar says 'Upgrade' with the subtitle 'Upgrade configuration and content from Nexus Repository Manager 2'. Below the title, there's a section titled 'Repositories' with the sub-instruction 'Select the repositories to be upgraded.' and 'Customize advanced configuration of the upgrade per-repository as needed.' A table lists 13 repositories, each with checkboxes for selecting them. The columns are: Repository, Type, Format, Supported, Status, Destination, Method, and a column of edit icons. Most repositories are checked. The table includes entries for Apache Snapshots, Central, Central M1 shadow, Local Development, npmjs Registry, NuGet Group, NuGet Gallery, NuGet Releases, Public Repositories, Releases, and Image Site. At the bottom of the table are buttons for 'Back', 'Next', and 'Cancel'.

Figure: Partial List of Repository Selections for Upgrade

Repositories

If *User-Managed Repositories* is one of your selections from the *Content* screen, the *Repositories* screen allows you to select which repositories you want to upgrade. As shown in *Figure: Partial List of Repository Selections for Upgrade*, you can select all repositories with one click, at the top of the table. Alternatively, you can select each individual repository and customize upgrade options for each repository with the configuration icons in the last column. In addition to *Repository*, the table displays information about the status of the repository. Keep in mind that the *Repository ID* defined in version 2 is displayed in the list and after the upgrade used as the *Name* of the repository.

Preview

This table displays a preview of the content set for the upgrade, selected in the previous screens. Click *Begin*, then confirm from the modal, that you want to start the upgrade process.

Running the Upgrade

After the upgrade was started in the *Preview* screen, the repository manager starts with a short *Preparing* step. From this point on, no further configuration changes should be performed on version 2. They will not be moved to version 3.

Any upgrade process invoked destroys any existing configuration in the target Nexus Repository Manager 3 server and replaces it with the upgraded configuration from version 2.

However, any content changes to the upgraded repositories continue to be upgraded during the following *Synchronizing* step. For example, new proxied components or new deployed components in version 2 are transferred to version 3.

During the transfer process, your content can already be viewed and accessed in version 3, for example via using the component search or browsing in repositories or repository groups. However, the repositories will be offline until the process is fully complete.

The status in the view shows the number of components transferred and when the last changes were detected in version 2. Once all components are migrated and no further changes have been detected for a while the upgrade is mostly complete. You can now decide upon waiting for further deployments (and component evaluations in the case of Nexus Firewall) to version 2 or finalizing the upgrade. To finalize, stop the monitoring and proceed through the *Finishing* screen.

After the Upgrade

With the upgrade completed and all components transferred, you can perform the next steps in your upgrade plan. These can include:

- Stopping Nexus Repository Manager 2.
- Archiving Nexus Repository Manager 2 and delete the install from the server.
- Reconfiguring Nexus Repository Manager 3 to use the HTTP port, context path and repository paths of version 2, if desired.
- Alternatively updating all tool configurations pointing to the repository manager, such as Maven settings files and POM files.

Configuring Legacy URL Paths

By default, Nexus Repository Manager 2 uses a different URL pattern to expose repositories and repository groups than Nexus Repository Manager 3. During upgrade from NXRM 2 to NXRM 3, a capability is added so that Nexus Repository Manager 3 automatically supports the old patterns and your automated tools and CI continue to work. This allows the example of `http://localhost:8081/nexus/repository/sample` to also be accessed using `http://localhost:8081/nexus/content/repositories/sample`.

⚠ Note

This example assumes your hostname (localhost), port (8081) and context path (nexus) match between your Nexus Repository Manager 2 and Nexus Repository Manager 3 installations. If not, you must utilize the ones from version 3 or reconfigure as stated in [Changing the Context Path section](#)(see page 189).

This capability has no affect on REST API calls because these were totally rewritten in NXRM3. Any REST API calls need to be reevaluated and rewritten against the NXRM3 [REST API](#)(see page 464).

Pre 3.7, if you want the old URL format, you must make a configuration change. This can be done in `$data-dir/nexus3/etc/nexus.properties` by adding:

```
org.sonatype.nexus.repository.httpbridge.internal.HttpBridgeModule.legacy=true
```

As with any Nexus Repository Manager configuration change, the server must be restarted for this to start working.

- ❗ Any automated tooling that uses direct repository browsing will not function in Nexus Repository Manager 3 as that is not currently supported.

Restarting an Upgrade

If you are running a Nexus Repository Manager 2 to 3 upgrade and it needs to be restarted for some reason (the upgrade was cancelled, or a problem occurred) it will be necessary to start the upgrade over from the beginning. To do this:

1. Shut down Nexus Repository Manager 2 and remove the entire NXRM 2 `$work-dir/db/migrationagent`⁶⁷⁶ directory.
2. Shut down Nexus Repository Manager 3 and remove the entire NXRM 3 `$data-dir`⁶⁷⁷ directory.

⁶⁷⁶<https://support.sonatype.com/hc/en-us/articles/213464308-Understanding-Nexus-Repository-Manager-2-Basic-Directories#WorkDir>

⁶⁷⁷<https://support.sonatype.com/hc/en-us/articles/218707647-How-to-Determine-the-Location-of-the-Nexus-3-Data-Directory>

User Interface

This section covers the basic aspects of the user interface of Nexus Repository Manager Pro and Nexus Repository Manager OSS including an overview of the user interface features, searching components and browsing repositories and other features that are, by default, available to anonymous users and similar read-only roles. Administrative tasks like configuring repositories, tasks, and security are documented in [Configuration](#)(see page 233).

Topics in this section:

- [User Interface Overview](#)(see page 209)
- [Searching for Components](#)(see page 213)
- [Browsing Repositories and Repository Groups](#)(see page 227)
- [Working with Your User Profile](#)(see page 229)
- [Uploading Components](#)(see page 230)
- [Viewing Tags](#)(see page 232)

User Interface Overview

The user interface is accessed via a web browser and works best with our [Recommended Browsers](#)(see page 146). The repository manager provides anonymous access for users who only need to search for components, browse the repositories and access components via client tools such as Maven or NuGet. This anonymous access level is a configurable, read-only mode that includes the main user interface elements as shown in *Figure: "User Interface for Anonymous Users"*.

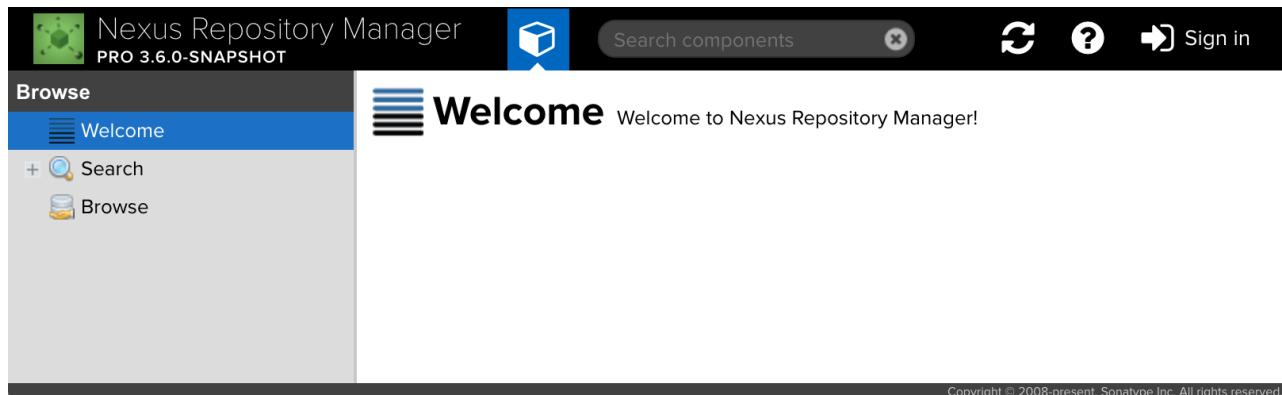


Figure: User Interface for Anonymous Users

Once a user is logged in further features become available depending on the user's privileges. An example for the admin user including the *Administration* menu icon is visible in *Figure: "User Interface for Logged In admin User"*.

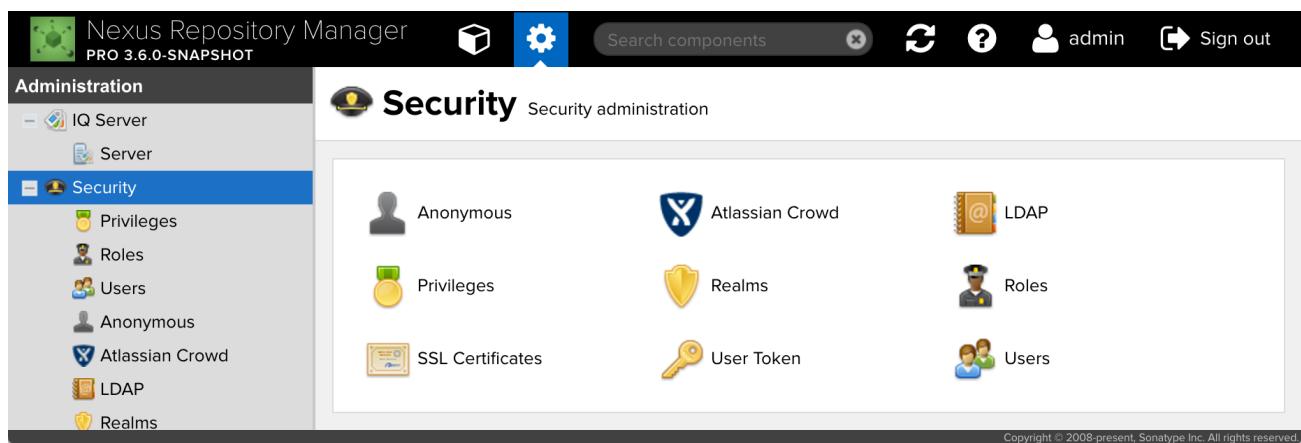


Figure: User Interface for Logged In admin User

The user interface is separated into a number of different sections.

Main toolbar

The top of the page contains the header with a number of elements starting on the left with the logo:

Logo and version label

The logo and the version label differ for Nexus Repository Manager OSS and Nexus Repository Manager Pro and allows you to know what version of the repository manager you are accessing at a glance.



Browse button

The browse button allows you to switch to the Browse menu items in the main menu section on the left of the user interface. The contents of the menu will depend on your assigned user privileges.



Administration button

The administration button allows to switch to the Administration menu items in the main menu section on the left of the user interface as visible in *Figure 3.2, "User Interface for Logged In admin User"*. The contents of the menu will depend on your assigned user privileges.

Search input box

The search input box can be used to start a keyword search. The results are displayed in the feature view panel.



Refresh button

The refresh button is a global refresh button that affects all views in the user interface including the feature view panel. E.g., it refreshes the search results view, the user list or the staging repository list, if they are currently the active feature view.

Help button

Clicking the help button opens up the help menu. It contains a link to specific help about the currently active feature view. The *About* item displays a dialog with details about the version as well as license and copyright information. The *Documentation*, *Knowledge base*, *Community*, *Issue tracker* and *Support* items link to the respective pages on the Sonatype websites.



The *Sign In* button allows you to sign in to the user interface as a specific user. Doing so gives you access to the privileges assigned to the user, changes the *Sign in* button to a *Sign out* button and adds a button displaying the user's name. The user's name button functions to access the Account feature view as part of the User menu in the main menu on the left with any other user features the account can access.

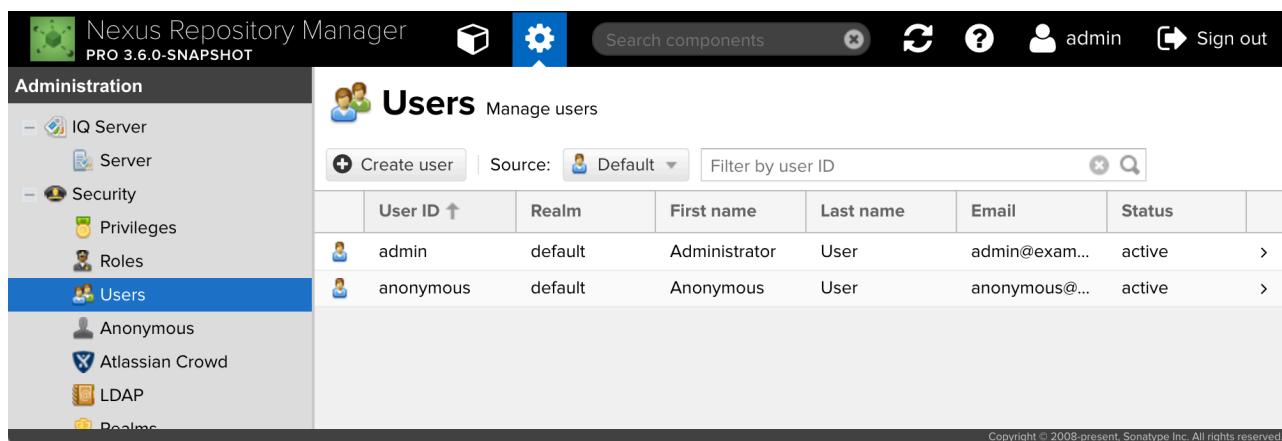
Main Menu

The main menu on the left contains either the Browse, the Administration or the User menu items. The exact list of available menu items depends on the current user's assigned privileges. E.g., the Administration menu as visible in *Figure: "User Interface for Logged In admin User"* includes the Security section, which is not available to anonymous users by default. The panel itself can be horizontally collapsed and expanded with the button in the top right-hand corner of the panel. Each submenu can be vertically collapsed and expanded with the button beside the title for each submenu. Selecting a menu item triggers the display of the respective feature view in the feature view panel.

Feature View Panel

The feature view panel in the center of the user interface right of the main menu initially displays the *Welcome* feature view. It changes display based on your selected item in the main menu.

Figure: "Typical Example Interface with a List" shows a typical user interface appearance of the repository manager with the Users feature view in the feature view panel. It shows a list of users.



Nexus Repository Manager
PRO 3.6.0-SNAPSHOT

Administration

- IQ Server
- Server
- Security
 - Privileges
 - Roles
 - Users**
 - Anonymous
 - Atlassian Crowd
 - LDAP
 - Realms

Users Manage users

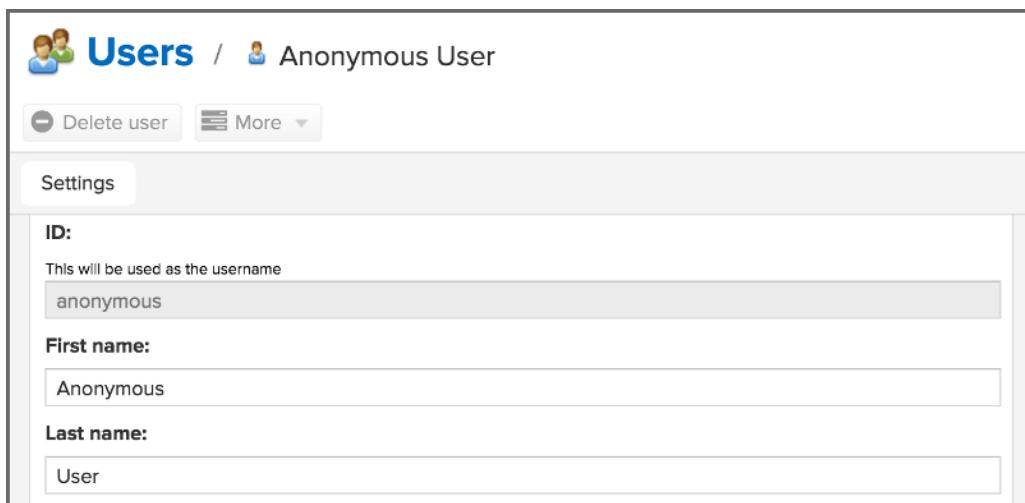
Create user | Source: Default | Filter by user ID

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

Copyright © 2008-present, Sonatype Inc. All rights reserved.

Figure: Typical Example Interface with a List

Clicking on a row in the list, switches the feature view to a specific display for the item in the row as visible in *Figure: "Typical Example Interface for Editing and Viewing"*. The top level navigation allows you get back to the list by clicking on the *Users* label. The form below has a number of sections that can be accessed via buttons as well as specific functionality like deletion and their associated buttons.



Users / **Anonymous User**

Delete user More

Settings

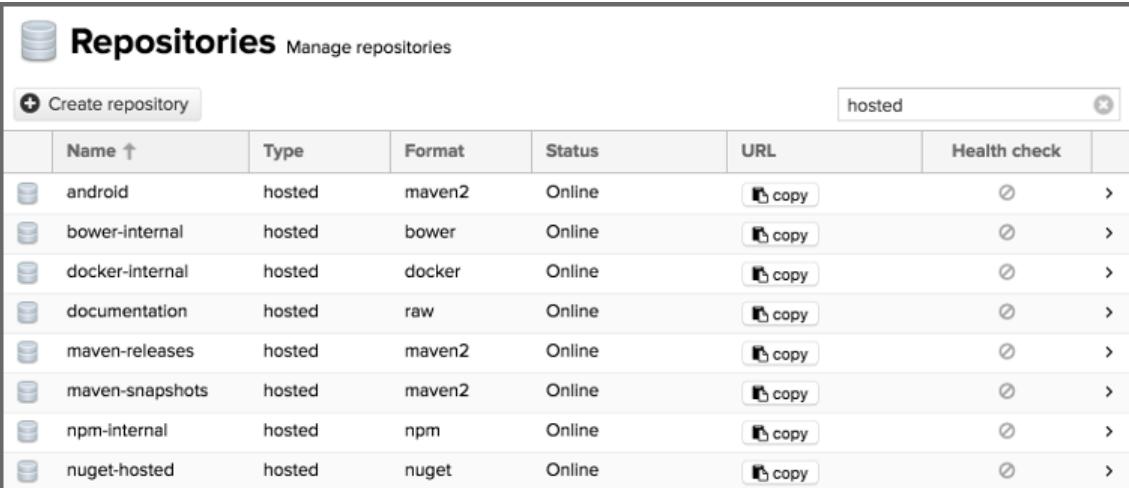
ID:
This will be used as the username

First name:

Last name:

Figure: Typical Example Interface for Editing and Viewing

The list header features buttons for various operations that differ per list as well as an input box that allows you to filter the list by any terms used in any column. *Figure: "Filtering the Repository List to Display Only Hosted Repositories"* shows an example use case where a user typed "hosted" in the filter box and the list of repositories only shows hosted repositories. This filtering works for all columns in a list and can be used in most list displays in the repository manager. For example, you can use it to filter the users list to find disabled users, filter the routing list, the roles list and others.



Name ↑	Type	Format	Status	URL	Health check	
android	hosted	maven2	Online	 copy		
bower-internal	hosted	bower	Online	 copy		
docker-internal	hosted	docker	Online	 copy		
documentation	hosted	raw	Online	 copy		
maven-releases	hosted	maven2	Online	 copy		
maven-snapshots	hosted	maven2	Online	 copy		
npm-internal	hosted	npm	Online	 copy		
nuget-hosted	hosted	nuget	Online	 copy		

Figure: Filtering the Repository List to Display Only Hosted Repositories

The column headers in most lists can be clicked to invoke a sorting of the list by the respective column as well as activate and deactivate some of the columns.

Searching for Components

 Available in Nexus Repository OSS and Nexus Repository Pro

Searching components in the repository manager is an important use case for being able to access information about specific components including different versions that are available, security and license data and other information as well as for build tool migrations, download of deployment packages and other component related development, QA and operations activities.

The different search modes can be accessed with the Browse button in the main toolbar and selecting Search or one of the nested options like Custom or Maven. The common feature view with the criteria drop-down selector for the search without results is displayed in *Figure: "Keyword Search with More criteria Input"*.

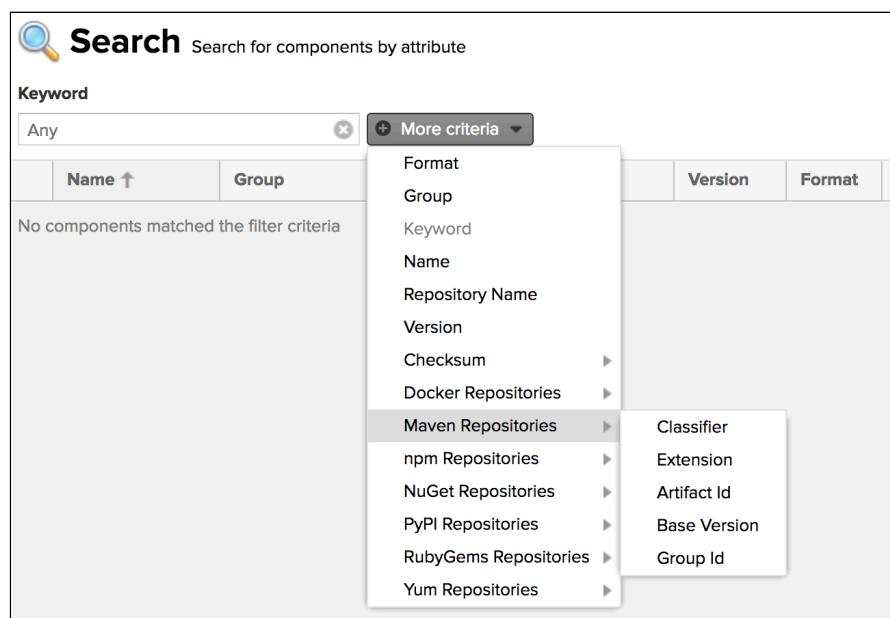


Figure: Keyword Search with More criteria Input

Beneath the search title is the search criteria input area that displays the current criteria input e.g., *Keyword*. Beside the current criteria is a More Criteria button that allows you to add further criteria to your search. Each criteria can be removed by clicking on the minus/dash icon within the criteria input box. The cross/x in the input box resets the value.

Each criteria can be used together to allow for broad or fine search results. For example, searching on name *foo* might return a large number of Maven, NuGet and other components but adding a version could limit it to what you're searching for. The Keyword field also supports the * (star, asterisk) character for wildcard matching. For example, a keyword search for `org.sonatype.nexus*` would return components containing `org.sonatype.nexus`, `org.sonatype.nexus.plugins` and any other matches. If nothing is specified in a criteria, the results have no reductions based around that criteria.

Search Criteria and Component Attributes

A number of criteria can be used with any repository format and returns results from all components in all repositories:

Keyword

A keyword is a string used for a search, where matches in *Format*, *Group*, *Name*, *Version* and all other component metadata values are returned.

Format

The format of the repository in which to look for a component.

Group

An identifier that groups components in some way, such as by organization. It can also be used to simply to create a specific namespace for a project. Not all repository formats use the notion of a group. Some tools simply use a different name for the concept e.g., org for Apache Ivy or groupId for Apache Maven and the maven2 repository format. In the case of a maven2 repository, group is a required attribute. Other formats, like the nuget repository format, do not use group at all.

Name

The name of a component constitutes its main identifier. Different repository formats use a different name for the concept such as artifactId for Apache Maven and the maven2 repository format.

Repository Name

The name of a repository in which to look for a component.

Version

The version of a component allows you to have different points in time of a component released. Various tools such as Maven or NuGet use the term version. Other build systems call this differently e.g. rev, short for revision, in the case of Apache Ivy. In most repository formats version numbers are not enforced to follow a specific standard and are simply a string. This affects the sort order and can produce unexpected results.

Checksum - MD5, SHA-1, SHA-256 or SHA-512

A checksum value of a component file generated by an MD5, SHA-1, SHA-256 or SHA-512 algorithm.

In addition there are criteria that can be used to search for components in repositories with specific formats only:

Conan Repositories

Base Version

Project or library version, for instance "1.0.0".

Channel

Represents an alternative variant of packages for the same library and usually describes the maturity of the package, like as "stable" or "testing". Also can show package revisions.

Common Version criteria for Conan format is combination of Base Version and Channel, for instance "1.0.0-stable"

Docker Repositories

Image Name

The name for the Docker image. It is equivalent to the *Name* of the component in the repository manager that represents the Docker image.

Image Tag

The tag for the Docker image. It is equivalent to the *Version* of the component in the repository manager that represents the Docker image.

Layer Id

The unique identifier for a Docker image layer.

Maven Repositories

Group Id

The Maven groupId for a component. Other build systems supporting the Maven repository format call this differently e.g. org for Apache Ivy and group for Gradle and Groovy Grape. Group Id is equivalent to *Group*.

Artifact Id

The Maven artifactId for a component. Other build systems call this differently e.g. name for Apache Ivy and Gradle, and module for Groovy Grape. Artifact Id is equivalent to *Name*.

Classifier

The Maven classifier for a component. Common values are javadoc, sources or tests.

Packaging

The Maven packaging for a component, which is jar by default. Other values as used in Maven and other build tools are ear, war, maven-plugin, pom, ejb, zip, tar.gz, aar and many others.

Base Version

The base version of the component/asset. Typically this is the same value as the version for release components. SNAPSHOT development components use a time-stamped version but the base version uses the SNAPSHOT version e.g. version of 1.0.0-20151001.193253-1 and base version of 1.0.0-SNAPSHOT.

Extension

The extension used for a specific asset of a component.

NuGet Repositories

ID

The NuGet component identifier is known as Package ID to NuGet users.

Tags

Additional information about a component formatted as space-delimited keywords, chosen by the package author.

p2 Repositories

Plugin Name

The name of a p2 plugin stored in NXRM.

PyPI Repositories**Classifiers**

Denote the maturity, intended audience, license and supported versions the creator wished associated with their component.

Description

Creator provided long description of the component.

PyPI Keywords

Associated component keywords. Generally used as identifiers to search.

Summary

Creator provided description of the component.

RubyGems Repositories**Platform**

The Platform the gem runs on defined via the gemspec.

Summary

A short summary of the gem's description defined via the gemspec.

Description

A long description of the gem defined via the gemspec. This field is optional when creating a gem so may be blank.

Yum Repositories**Package Name**

The name of the package, this field is the equivalent of *Name* for Yum.

Architecture

The architecture the package is designed to be run on.

Keyword search query string handling

The query you enter is passed into the indexes made against every component in the system. These indexes contain the component coordinates as well as any metadata attached to the component. Some of these indexes contain data that has been tokenized (for example, strings split on a hyphen, so `aether-util` would match 2 search terms, `aether` or `util`), while other indexes do not (imagine wanting to search for "aether-util" (see below for more details on double quotes), this would require the full name to match against, rather

than a tokenized version). Without wildcards, your query string must match a field in one of these indexes exactly, wildcards can save you when you just can't remember the exact name, but have some partial idea.

About wildcards, how do they change my queries?

Let's suppose you did a query for `aeth`, and it returned no results, as the component you are trying to match needs the full term `aether` to make a match. So you can then use `aeth*` and this would now match the `aether-util` component you were initially searching for. Keep in mind that you can use wildcards anywhere in the query string, but placed anywhere except for the last character position has potential to cause the query to run slow.

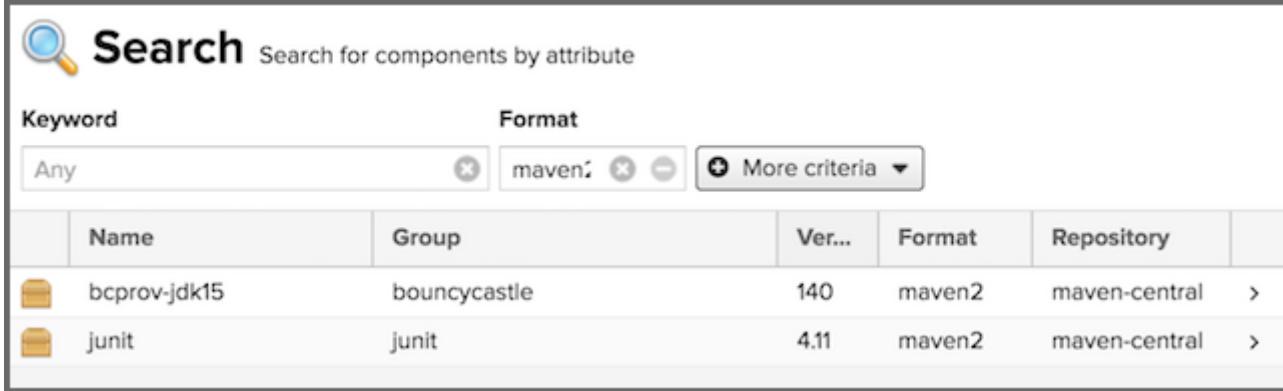
Using a hyphen in my query generates a lot of results, how can I make this more manageable?

Along with tokenizing strings for search terms in some of the indexes, we also do the same to the query string you type, so if you type `aether-util` we will show all results that would match `aether` or `util`. Suppose you know you want exactly `aether-util` and want to filter out all the other results that were brought in, you can use double quotes in your search term, "`aether-util`" for example, this would then no longer tokenize your query string, but require the full string for a match.

Search Results

Once you have provided your search terms in one or multiple criteria input fields, like the Keyword criteria in the Search feature view, the first 1000 results become visible in the component list, with an example displayed in *Figure: "Results of a Component Search for format maven2"*. If more than 1000 results exist, notation will appear under the filters relaying how many items were found in total.

The components are listed with their *Name*, *Group*, *Version*, *Format* and *Repository* information and by default are sorted alphabetically by *Name*. Columns and sort order can be adjusted like in all other lists.



The screenshot shows the 'Search' feature view. At the top, there is a search bar with a magnifying glass icon and the text 'Search for components by attribute'. Below the search bar, there are two input fields: 'Keyword' (containing 'Any') and 'Format' (containing 'maven2'). To the right of these fields is a button labeled 'More criteria' with a dropdown arrow. Below the search controls is a table displaying search results. The table has columns: Name, Group, Ver..., Format, and Repository. There are two rows of data:

	Name	Group	Ver...	Format	Repository	
	bcprov-jdk15	bouncycastle	140	maven2	maven-central	>
	junit	junit	4.11	maven2	maven-central	>

Figure: Results of a Component Search for format maven2

Selecting a component in the list changes to a display of the component information documented in [Viewing Component Information](#)(see page 221).

Preconfigured Searches

Keyword Search

The main toolbar includes a Search components text input field. Type your search term and press enter and the repository manager performs a search by *Keyword*.

The same search can be accessed by selecting the Search item in the *Browse* main menu. The search term can be provided in the *Keyword* input field in the *Search* feature view.

Custom Search

A configurable search using the criteria you select is available via the *Custom* menu item in the *Search* section of the *Browse* main menu. Initially it has no criteria and it allows you to create a search with criteria you add with the *More Criteria* button.

Apt Search

The Apt search is a predefined search available via the *Apt* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Package Name* and *Version* and supports adding further criteria. The format is configured to *apt*.

Bower Search

The Bower search is a predefined search available via the *Bower* menu item in the *Search* section of the *Browse* menu. It defaults to inputs for *Name* and *Version* and supports adding further criteria. The format is configured to *bower*.

CocoaPods Search

The CocoaPods search is a predefined search available via the *CocoaPods* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Package Name* and *Version* and supports adding further criteria. The format is configured to *cocoapods*.

Conan Search

The Conan search is a predefined search available via the *Conan* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Package Name*, *Base Version* and *Channel* and supports adding further criteria. The format is configured to *conan*.

Conda Search

The Conda search is a predefined search available via the *Conda* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Package Name* and *Version* and supports adding further criteria. The format is configured to *conda*.

Docker Search

The Docker search is a predefined search available via the *Docker* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Image Name*, *Image Tag* and *Layer Id* and supports adding further criteria. The format is configured to *docker*.

Git LFS Search

The Git LFS search is a predefined search available via the *Git LFS* menu item in the *Search* section of the *Browse* main menu. It defaults to an input for Name and supports adding further criteria. The format is configured to *gitlfs*. Note that since Git LFS is not provided with the original filename for a tracked file, the Name field actually contains the generated OID which can be found in the corresponding pointer file's contents.

Helm Search

The Helm search is a predefined search available via the *Helm* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Name* and *Version* and supports adding further criteria. The format is configured to *helm*.

Maven Search

The Maven search is a predefined search available via the *Maven* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Group Id*, *Artifact Id*, *Version*, *Base Version*, *Classifier* and *Extension* and supports adding further criteria. The format is configured to *maven2*.

NuGet Search

The NuGet search is a predefined search available via the *NuGet* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *ID* and *Tags* and supports adding further criteria. The format is configured to *NuGet*.

npm Search

The npm search is a predefined search available via the *npm* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Scope*, *Name*, and *Version* and supports adding further criteria. The format is configured to *npm*.

p2 Search

The p2 search is a predefined search available via the *p2* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Package Name* and *Plugin Name* and supports adding further criteria. The format is configured to *p2*.

PyPI Search

The PyPI search is a predefined search available via the *PyPI* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Classifiers*, *Description*, *PyPI Keywords*, and *Summary* and supports adding further criteria. The format is configured to *pypi*.

R Search

The R search is a predefined search available via the *R* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Package Name* and *Version* and supports adding further criteria. The format is configured to *r*.

Raw Search

The Raw search is a predefined search available via the *Raw* menu item in the *Search* section of the *Browse* main menu. It defaults to an input for *Name* and supports adding further criteria. The format is configured to *raw*.

RubyGems Search

The RubyGems search is a predefined search available via the *RubyGems* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Name*, *Version*, *Platform*, *Summary* and *Description* and supports adding further criteria. The format is configured to *rubygems*.

Yum Search

The Yum search is a predefined search available via the *Yum* menu item in the *Search* section of the *Browse* main menu. It defaults to inputs for *Package Name*, *Version* and *Architecture* and supports adding further criteria. The format is configured to *yum*.

Example Use Case - SHA-1 Search

Sometimes it is necessary to determine the version of a component, where you only have access to the binary file without any detailed component information. When attempting this identification and neither the filename nor the contents of the file contain any useful information about the exact version of the component, you can use SHA-1 search to identify the component.

Create a sha1 checksum, e.g., with the `sha1sum` command available on Linux or OSX or `fciv` on Windows, and use the created string in a Custom search by adding the SHA-1 criteria from the Checksum section of the More criteria control.

The search will return a result, which will provide you with the detailed information about the file allowing you to replace the file with a dependency declaration. E.g. you can derive the Maven coordinates of a jar file and use them in a dependency declaration.

A SHA-1 or similar checksum search can be a huge timesaver when migrating from a legacy build system, where the used libraries are checked into the version control system as binary components with no version information available.

Viewing Component Information

Once you located a component via a search and selected it in the list, you see the component information. An example is displayed in *Figure: "Example for Component Information and List of Associated Assets"*.

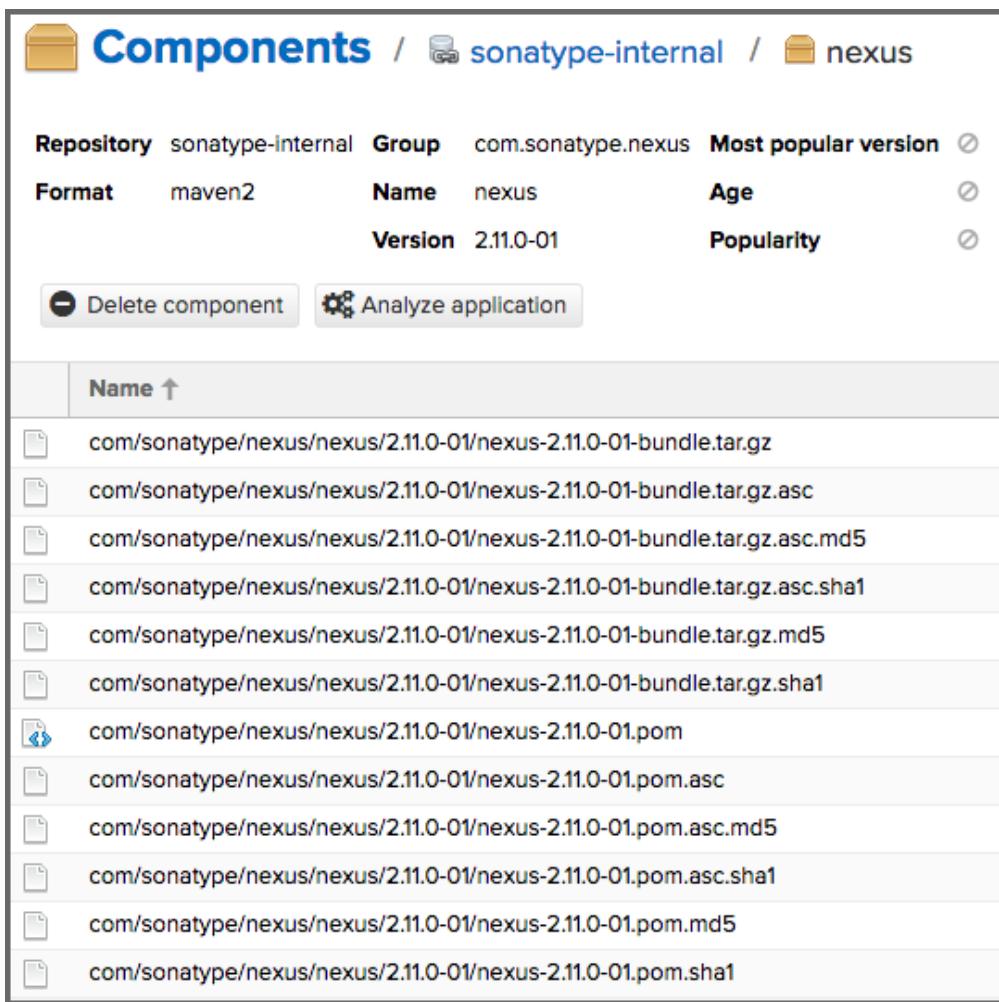
The information displayed includes the name and format of the repository that contains the component as well as the component identifiers *Group*, *Name*, and *Version*. *Most popular version* contains the version number of the same component that is most popular in its usage within a specific group and name.

Popularity shows a relative percentage of popularity between the displayed component against all other versions of this component. A value of 100% signals this version to be the most popular. 50% means that the specific version is half as popular as the most popular version. Popularity data is provided by the Sonatype

Data Services based on requests from the Central Repository and other data and not available for all components. Age shows the age of the component.

None of the popularity or age data is viewable without Repository Health Check enabled, as lightly touched on in [Managing Repositories and Repository Groups](#)(see page 240).

A list of one or more assets associated with the component is shown below the component information. Click on the row with the Name of the asset you want to inspect to view the asset information documented in [Viewing Asset Information](#)(see page 223).



The screenshot shows the 'Components' section of the Nexus Repository Manager. At the top, there are navigation links: 'Components' (with a folder icon), 'sonatype-internal' (with a database icon), and 'nexus' (with a folder icon). Below this is a table with the following columns and data:

Repository	sonatype-internal	Group	com.sonatype.nexus	Most popular version	Age	Popularity
Format	maven2	Name	nexus	2.11.0-01	2.11.0-01	2.11.0-01

Below the table are two buttons: 'Delete component' and 'Analyze application'. The main area displays a list of associated assets, each with a small file icon and a link to its details:

- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01-bundle.tar.gz
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01-bundle.tar.gz.asc
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01-bundle.tar.gz.asc.md5
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01-bundle.tar.gz.asc.sha1
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01-bundle.tar.gz.md5
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01-bundle.tar.gz.sha1
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01.pom
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01.pom.asc
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01.pom.asc.md5
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01.pom.asc.sha1
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01.pom.md5
- com/sonatype/nexus/nexus/2.11.0-01/nexus-2.11.0-01.pom.sha1

Figure: Example for Component Information and List of Associated Assets

To delete a component press the *Delete component* button as shown in *Figure: "Example for Component Information and List of Associated Assets"*. A modal will pop up to confirm the deletion. You can only delete components from hosted and proxy repositories. A deletion of a components triggers the deletion of all its associated assets, in most repository formats.

In some repository formats assets are shared across components. They remain after a component deletion. For example, while a Docker image is a component and can be deleted, the layers that make it up remain after its deletion as these assets are potentially shared with other Docker images.

Analyze Application

Application analysis performs a deep inspection of this application, identifying potential risks. More information is available [here](#).

Email address:
The address where the summary report will be sent

Report password:
A password to gain access to the detailed report

Proprietary packages:
A comma separated list of proprietary packages

Report label:
The name the report will be given

Analyze **Cancel**

Figure: Analyze Application Form

To analyze an application, press the *Analyze application* button as shown in *Figure: "Example for Component Information and List of Associated Assets"*. A form will pop up to request further information from you: email address, report password, a list of proprietary packages for the application, and a name for the report. Once you provide this information, press the *Analyze* button as shown in *Figure: "Analyze Application Form"*. Your report link will be emailed to you as soon as it is finished.

Viewing Asset Information

Available in Nexus Repository OSS and Nexus Repository Pro

Asset information can be accessed from a component information view. The *Delete* button allows you to remove an asset. The information itself is broken up into sections, accessible by tabs below the *Delete Asset* button. The Summary section contains a number of attributes about the specific asset. An example for search is displayed in *Figure: "Asset Info Example"*.

Path:

the path to the asset in the repository. This is a link, either downloadable or loading in browser dependent on the contents of the asset.

Content type:

the MIME type of the asset

File size:

the size of the file

Blob created:

the date and time when the asset was created in the Nexus Repository Manager blob store

Blob updated:

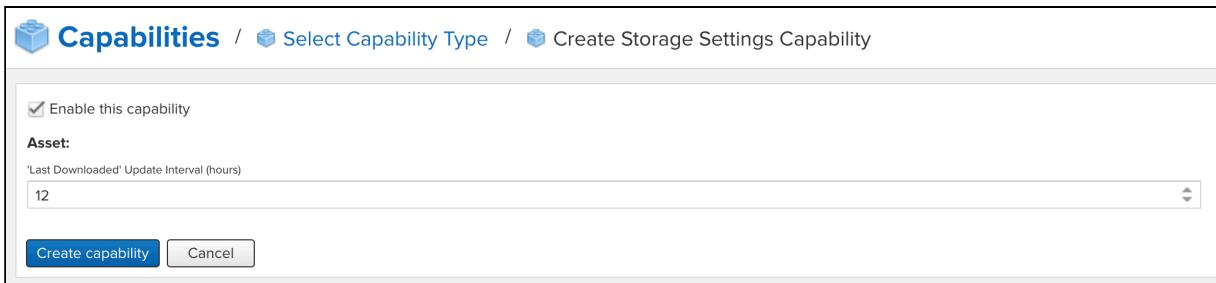
the date and time when the asset was updated last in the Nexus Repository Manager blob store. This will initially match *Blob created*.

Last 30 days:

the number of times the asset was downloaded in the last thirty days

Last downloaded:

the date when the asset was last downloaded (or updated). It will be updated when the asset is first downloaded and then on subsequent 200 and 304 responses. By default, this is prevented from updating too often for performance reasons and will only update once in any given 12 hour period. It is possible to override this setting by adding a *Storage Settings* capability (shown below) and configuring the '*Last Downloaded*' *Update Interval* value.



The screenshot shows a user interface for creating a storage settings capability. At the top, there's a breadcrumb navigation: 'Capabilities' / 'Select Capability Type' / 'Create Storage Settings Capability'. Below this, there's a checked checkbox labeled 'Enable this capability'. Under the 'Asset' section, there's a field labeled "'Last Downloaded' Update Interval (hours)' with the value '12' entered. At the bottom of the dialog are two buttons: 'Create capability' (in blue) and 'Cancel'.

Locally cached:

value of true means the asset can be found in the repository manager storage while false indicates that the metadata about the asset is available, though the asset itself has not been downloaded

Blob reference:

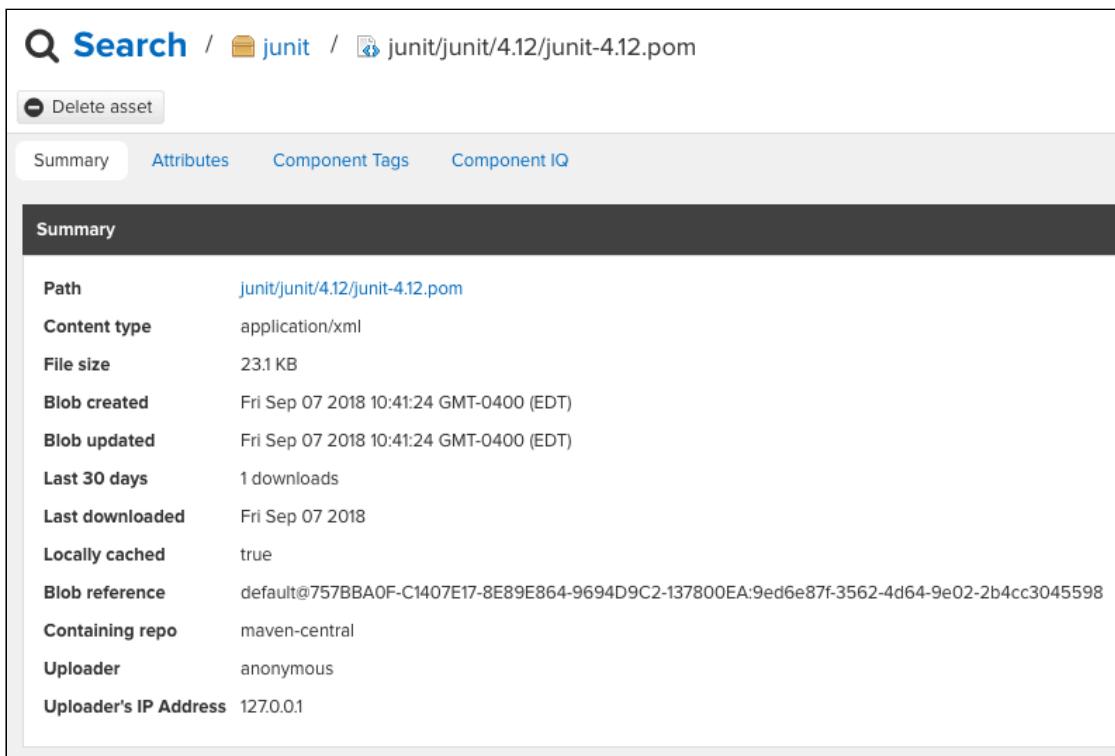
a unique identifier pointing at the binary blob representing the asset in the repository manager storage

Uploader:

user who uploaded the asset

Uploader's IP Address:

the IP address of the individual who uploaded or requested the asset



Path	junit/junit/4.12/junit-4.12.pom
Content type	application/xml
File size	23.1 KB
Blob created	Fri Sep 07 2018 10:41:24 GMT-0400 (EDT)
Blob updated	Fri Sep 07 2018 10:41:24 GMT-0400 (EDT)
Last 30 days	1 downloads
Last downloaded	Fri Sep 07 2018
Locally cached	true
Blob reference	default@757BBA0F-C1407E17-8E89E864-9694D9C2-I37800EA:9ed6e87f-3562-4d64-9e02-2b4cc3045598
Containing repo	maven-central
Uploader	anonymous
Uploader's IP Address	127.0.0.1

Figure: Asset Info Example

The *Attributes* section contains further metadata about the asset related to *Cache*, *Checksum*, and *Content* attributes. An example is displayed in *Figure: “Asset Attributes Example”*. Assets can include format specific attributes displayed in additional sections. For example an asset in a Maven2 repository has a *Maven2* section with attributes for *extension*, *baseVersion*, *groupId*, *artifactId*, *version*, and *asset_kind*.

Search /  junit /  junit/junit/4.12/junit-4.12.pom	
<input type="button" value="Delete asset"/>	
Summary	Attributes
Cache	
last_verified	2018-09-07T10:41:24.104-04:00
Checksum	
sha1	35fb238baee3f3af739074d723279ebea2028398
md5	af7ca61fba26556cf5b40cf15aad14
Content	
last_modified	2014-12-04T11:17:43.000-05:00
etag	af7ca61fba26556cf5b40cf15aad14
Maven2	
extension	pom
artifactId	junit
baseVersion	4.12
version	4.12
asset_kind	ARTIFACT
groupId	junit
Provenance	
hashes_not_verified	

Figure: Asset Attributes Example

In Nexus Repository Manager Pro, two additional tabs named *Component Tags* and *Component IQ* are available. The *Component Tags* tab contains a list of tag names (*firstCreated* and *lastUpdated* dates) associated with the selected component. When a row in this table is clicked the user will be presented with the tag details. An example is displayed in *Figure: "Component Tags Example"*.

Search /  junit /  junit/junit/4.12/junit-4.12.jar		
<input type="button" value="Delete asset"/>		
Summary	Attributes	Component Tags
Tag Name ↑	First Created Time	Last Updated Time
build-123	9/27/2018, 10:37:04 AM	9/27/2018, 10:37:04 AM

Figure: Component Tags Example

If Repository Health Check is enabled or Nexus IQ Server is connected, the *Component IQ* tab shows security information and license details about a component, if available to that format and from Sonatype.

Browsing Repositories and Repository Groups

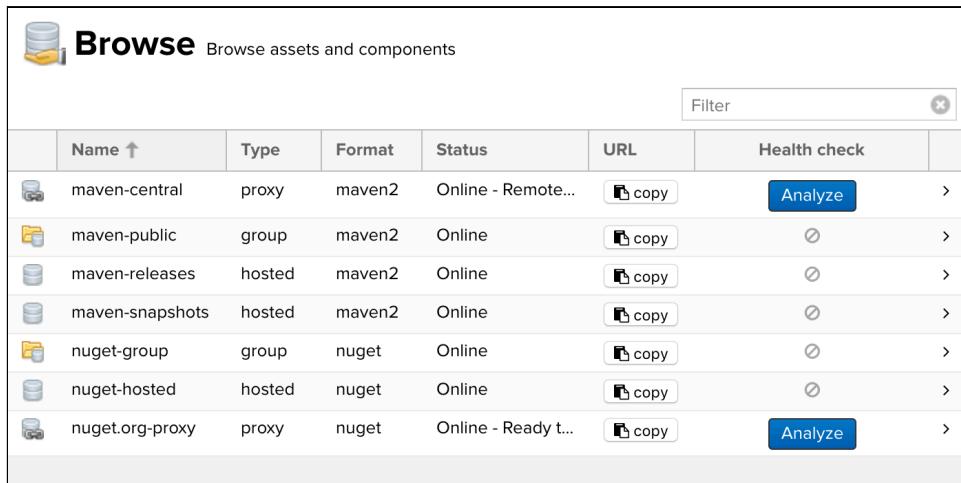
 **Available in Nexus Repository OSS and Nexus Repository Pro**

One of the most straightforward uses of the repository manager is to browse. Browsing allows you to inspect the contents of any repository or repository group for all the supported repository formats.

Click on the Browse



button in the main toolbar to access the Browse feature. This feature allows you to select a repository or repository group to browse from the list of all repositories as displayed in *Figure: "List of Repositories to Access for Component Browsing"*.



Browse <small>Browse assets and components</small>						
	Name 	Type	Format	Status	URL	Health check
	maven-central	proxy	maven2	Online - Remote...	 copy	
	maven-public	group	maven2	Online	 copy	
	maven-releases	hosted	maven2	Online	 copy	
	maven-snapshots	hosted	maven2	Online	 copy	
	nuget-group	group	nuget	Online	 copy	
	nuget-hosted	hosted	nuget	Online	 copy	
	nuget.org-proxy	proxy	nuget	Online - Ready t...	 copy	

Figure: List of Repositories to Access for Component Browsing

Once you click on the row for a specific repository a tree containing all assets in the repository is displayed. You can filter the tree content, expand nodes you're interested in, and select components or assets for more detail. Nodes in the tree are sorted in case insensitive order. Versions are sorted semantically. This tree is shown in *Figure: "Tree View Containing All Assets in a Repository"*.

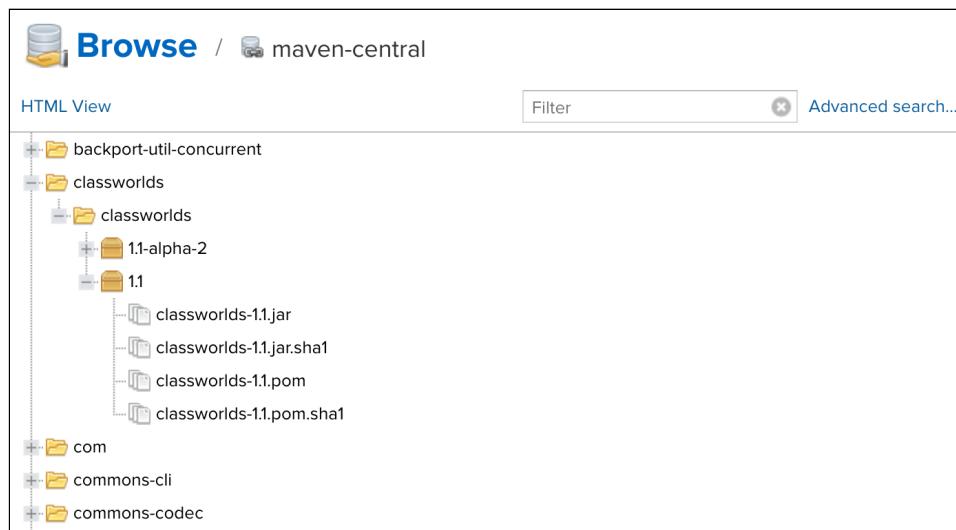


Figure: Tree View Containing All Assets in a Repository

When you select a component or asset, an information view appears on the right. This view is shown in Figure: “*Information View for a Selection in the Tree View*”. The information in this view is the same as the asset and component information views for search, but laid out as a series of scrolling panels instead of separate tabs. Additionally, in Nexus Repository Manager Pro there will also be a scrolling panel named *Component Tags* that will present tags information in the same manner as when viewing tag information in the [search results view](#)(see page 213).

Summary	
Repository	maven-central
Format	maven2
Component Group	classworlds
Component Name	classworlds
Component Version	1.1
Path	classworlds/classworlds/1.1/classworlds-1.1.jar
Content type	application/java-archive
File size	37 KB
Blob created	Wed Oct 18 2017 12:01:33 GMT-0700 (PDT)
Blob updated	Wed Oct 18 2017 12:01:33 GMT-0700 (PDT)

Figure: Information View for a Selection in the Tree View

Working with Your User Profile

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

As a logged-in user, you can click on your user name on the right-hand side of the main toolbar to switch the main menu to contain the *User* menu. Pressing on the *Account* menu item displays the *Account* feature in the main feature panel as displayed in *Figure: "Editing User Details in the Account Feature Panel"*.

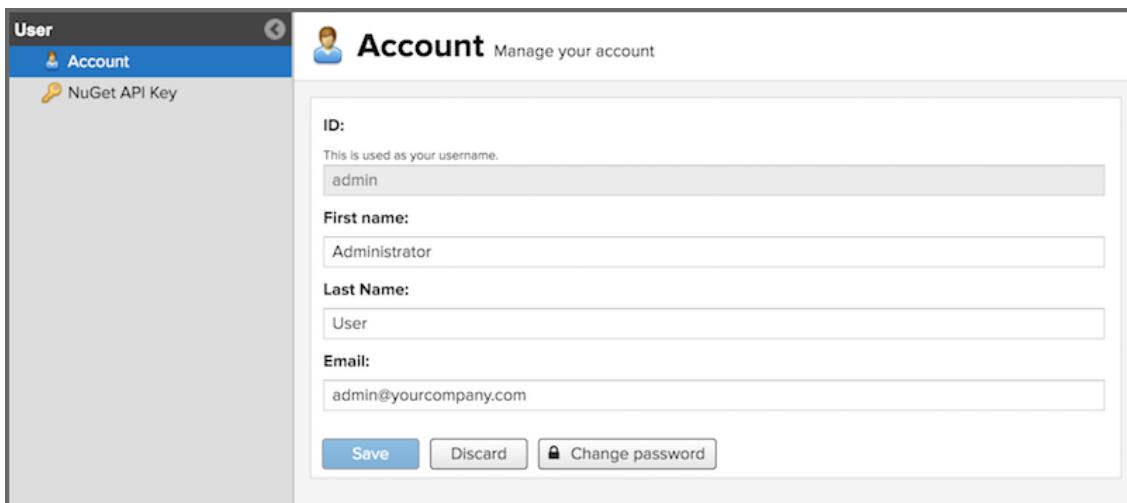


Figure: Editing User Details in the Account Feature Panel

The *Account* feature allows you to edit your *First Name*, *Last Name*, and *Email* directly in the form.

Changing Your Password

In addition to changing your name and email, the user profile allows you to change your password by clicking on the *Change password* button. You will be prompted to authenticate with your current password and subsequently supply your new password in pop up dialogs.

-  The password change feature only works with the built-in security realm. If you are using a different security realm like LDAP or Crowd, this option will not be visible.

Uploading Components

When your build makes use of proprietary or custom dependencies that are not available from public repositories, you will often need to find a way to make them available to developers in a custom repository. Nexus Repository Manager makes it easy to upload these third-party components to any of your hosted repositories via the UI, as shown below.

Before you begin

There are a few things you need before you can use this feature:

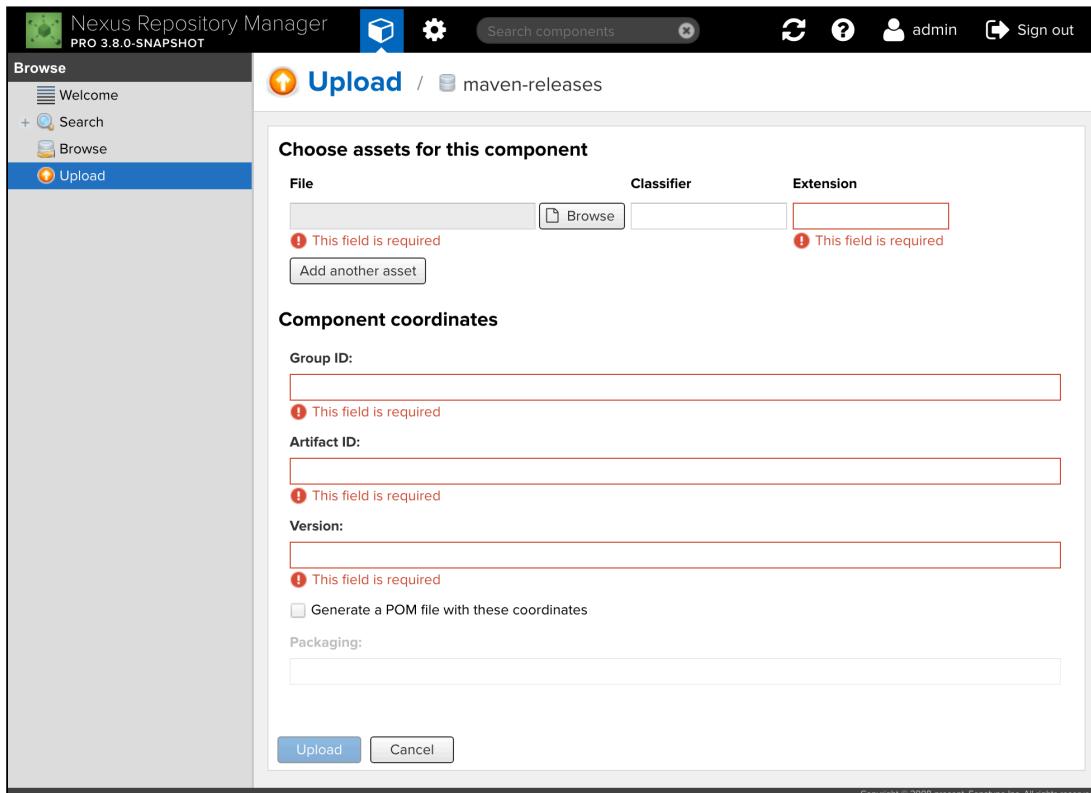
1. The *nx-component-upload* [privilege](#)(see page 313) assigned to the user (*anonymous* users do not have this privilege by default, but *admins* do)
2. One or more *hosted* repositories (only formats listed in the table below are supported)
3. Users must have the privilege to edit the repository. It is worth noting that if a user has *browse* and *read* privileges, the user will still see the *Upload* button, however an message will be displayed that they are not authorized if they try to upload a file.
4. One or more files of the proper type(s) for the intended repository format.

Valid file types for each format are enumerated here:

Repository format	Valid file types	Multiple file upload
Apt	.deb, .udeb	No
Maven (release only)	No restrictions, but common types include .pom, .jar, and .zip	Yes
Helm	.tgz, .tgz.prov	No
npm	.tgz	No
NuGet	.nupkg	No
PyPI	.tar.gz	No
R	.tar.gz, .tgz	No
Raw	No restrictions	Yes
RubyGems	.gem	No
Yum	None with permissive policy, otherwise .rpm, and .drpm	No

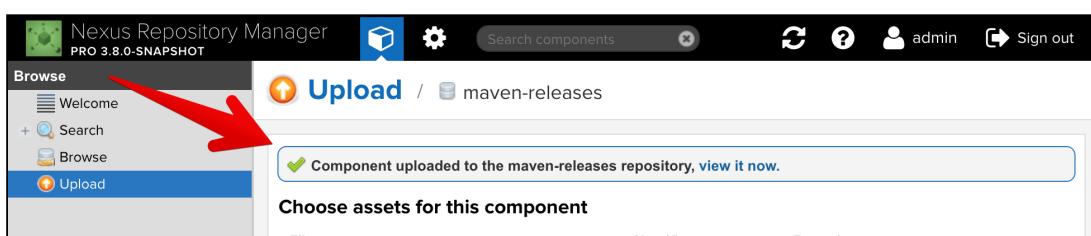
How to upload a component

To upload components to a repository, select a hosted repository in the *Browse* feature and then click on the *Upload component* button, which will display the screen shown below.



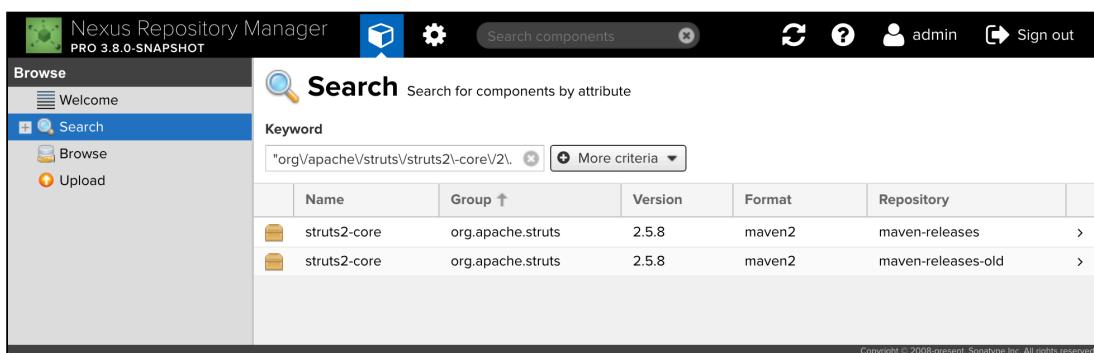
The screenshot shows the 'Upload' interface for the 'maven-releases' repository. The 'File' and 'Extension' fields are required and highlighted in red. The 'Group ID', 'Artifact ID', and 'Version' fields are also required and highlighted in red. The 'Upload' button is visible at the bottom.

The details depend on the repository in question, but to upload a component, fill out all of the required fields and click on the *Upload* button. Once the upload completes successfully, the form will be cleared and a message appears with a link to the component you uploaded, as seen here:



The screenshot shows the 'Upload' interface for the 'maven-releases' repository after a successful upload. A message at the top says 'Component uploaded to the maven-releases repository, [view it now.](#)'. A red arrow points to the 'Search' link in the left sidebar.

If you click this link, you will be taken to the *Search* feature where you can further inspect the component you uploaded. This is a cross-repository search, so you can easily detect whether an identical component already exists in another repository. This is demonstrated here:



Name	Group ↑	Version	Format	Repository
struts2-core	org.apache.struts	2.5.8	maven2	maven-releases >
struts2-core	org.apache.struts	2.5.8	maven2	maven-releases-old >

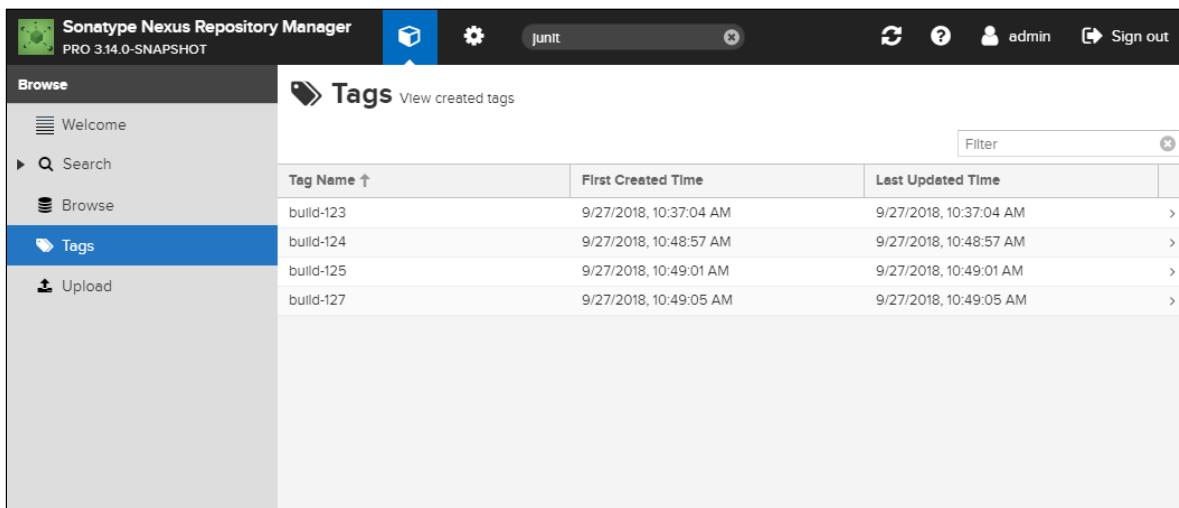
Advanced usage

If you have an advanced use case that isn't supported by this feature, such as batch uploads or automatically extracting coordinates from a .jar file, look into the component endpoint of our [REST and Integration API](#)(see page 464).

Viewing Tags

 Available in Nexus Repository Manager Pro only

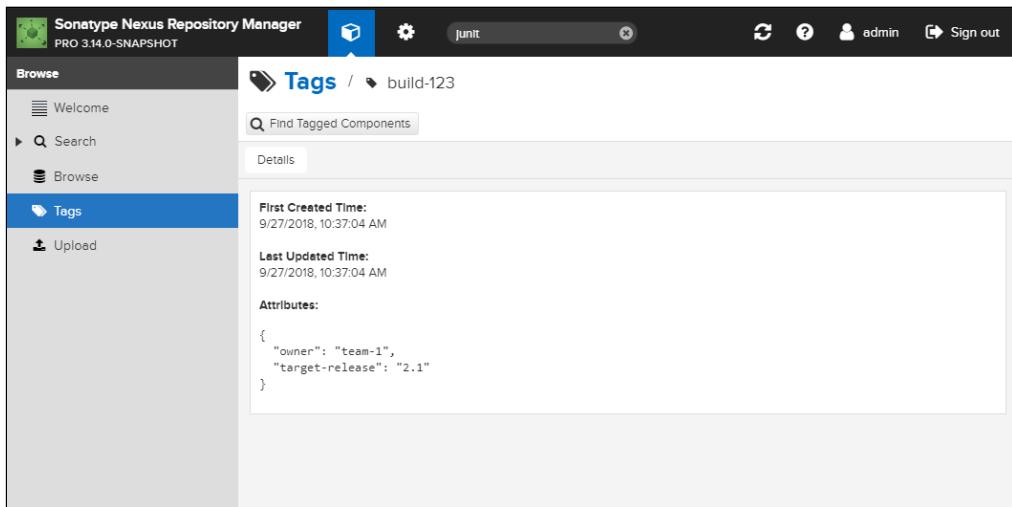
Tags that have been [created\(see page 357\)](#) via the REST API can be viewed through the user interface through the *Tags* entry on the sidebar if the active user has the *nexus:tags:read* permission.



Tag Name ↑	First Created Time	Last Updated Time
build-123	9/27/2018, 10:37:04 AM	9/27/2018, 10:37:04 AM
build-124	9/27/2018, 10:48:57 AM	9/27/2018, 10:48:57 AM
build-125	9/27/2018, 10:49:01 AM	9/27/2018, 10:49:01 AM
build-127	9/27/2018, 10:49:05 AM	9/27/2018, 10:49:05 AM

List of tags

Clicking on a row in the table will show a view where the attributes of a tag can be viewed.



Details of selected tag

Clicking the *Find Tagged Components* button will open [search\(see page 213\)](#) with a query for the tag currently viewed.

Configuration

This section covers all aspects of configuring Nexus Repository Manager Pro and Nexus Repository Manager OSS. Specifically the sections and menu items of the Administration main menu are covered. It can be accessed by authorized users by pressing the Administration button in the main toolbar.

-  The default user for accessing these features has the username *admin* and the password is configured as part of [initial setup\(see page 190\)](#). More fine-grained access can be configured as detailed in [Security\(see page 309\)](#).

Topics in this section:

- [Administration Menu\(see page 234\)](#)
- [Repository Management\(see page 234\)](#)
- [Support Features\(see page 254\)](#)
- [System Configuration\(see page 258\)](#)
- [License Management\(see page 280\)](#)
- [Storage Guide\(see page 281\)](#)
- [Auditing\(see page 286\)](#)
- [Routing Rules\(see page 287\)](#)

- [Retry Limit Configuration](#)(see page 289)

Administration Menu

The *Administration* menu, located on the left panel, contains the following sections:

Repository

The *Repository* section allows you to manage all Repositories and related configurations such as Routing and Targets.

IQ Server

The *IQ Server* item allows you to configure the connection of Nexus Repository Manager to Nexus IQ Server. Further documentation is available in the [Nexus IQ Server documentation](#)⁶⁷⁸.

 If you desire to utilize Nexus Firewall to quarantine and block unacceptable components that may harm your repository manager, review the [quick start guide](#)⁶⁷⁹.

Security

This section provides access to all the configuration features related to authentication and authorization of users including *Privileges*, *Roles*, *Users*, but also *LDAP*, *Atlassian Crowd*, *SSL Certificates* and *User Token*.

Support

Access a number of features that allow you to administer and monitor your repository manager successfully like *Logging* and *System Information*.

System

The general configuration for getting started and running the repository manager with e.g., *HTTP* or *Email* Server settings, but also *Capabilities* and *Tasks* to run regularly and other configurations.

Repository Management

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

⁶⁷⁸ <https://help.sonatype.com/display/NXIQ/IQ+Server>

⁶⁷⁹ <https://help.sonatype.com/display/NXIQ/Nexus+Firewall+Quick+Start>

Repositories are the containers for the components provided to your users as explained in more detail in [Repository Manager Concepts](#)(see page 160). Creating and managing repositories is an essential part of your Nexus Repository Manager configuration, since it allows you to expose more components to your users. It supports proxy repositories, hosted repositories and repository groups using a number of different repository formats.

The binary parts of a repository are stored in blob stores, which can be configured by selecting *Blob Stores* from the *Repository* sub menu of the *Administration* menu.

To manage repositories select the *Repositories* item in the *Repository* sub menu of the *Administration* menu.

Blob Stores

A blob store is the internal storage mechanism for the binary parts of components and their assets. They can be local file system or cloud-based Amazon S3. Each blob store can be used by one or multiple repositories and repository groups. You can learn more about blob stores in the [Storage Guide](#)(see page 281). Licensed Professional users may be interested in [group blob stores](#)(see page 284).

A *default* blob store that is based on a file system storage within the `$data-dir` is automatically configured.

The *Blob Stores* feature view available via the *Blob Stores* item in the *Repository* sub menu of the *Administration* menu displays a list of all configured blob stores. The columns provide some detail about each blob store:

Name

the name of the blob store as displayed in the repository administration

Type

the type of the blob store backend. *File* represents a file system-based storage and *S3* allows blobs to be stored in AWS S3 cloud storage. *Group* allows licensed Professional instances to take a combination of *File* and *S3* based blob stores to use in a rotation. Group blob stores are covered in more depth in a [subsection of the Storage Guide](#)(see page 284).

State

the state of the blob store; *Started* indicates it is running as expected, *Failed* indicates that there is a configuration issue and as a consequence, the blob store failed to initialise

Blob count

the number of blobs currently stored

Total size

the size of the blob store

Available space

the overall space available for the blob store

Click on a specific row to inspect further details of the selected blob store. The details view displays *Type* and *Name* and the absolute *Path* to the file system storage.

The *Create blob store* button allows you to add further blob stores. You can configure the *Type* and *Name* for the blob store. The *Path* parameter should be an absolute path to the desired file system location. It has to be fully accessible by the operating system user account running the Nexus repository manager.

Once a blob store has been created it can no longer be modified and any blob store used by a repository or repository group can not be deleted.

Blobs deleted in a repository are only marked for deletion. The *Compact blob store task*([see page 269](#)) can be used to permanently delete these soft-deleted blobs and therefore free up the used storage space.

Choosing the Number of Blob Stores

You will need to choose how many blob stores to create, and how you allocate repositories to these blob stores. This decision should be based on:

- the size of your repositories
- the rate at which you expect them to grow over time
- the storage space available to your Nexus Repository Manager
- the options you have available for adding storage space

For the time being, once a repository is allocated to a blob store, it is there permanently. Blob stores can be moved from one storage device to another (e.g. to a larger storage device) using a manual process, but blob stores cannot be split, nor can repositories span multiple blob stores. For these reasons, your approach to using blob stores should be chosen carefully.

The simplest approach is to create a single blob store per storage device and divide your repositories among them. This is suitable if either:

- Your repositories are growing slowly enough that you won't exceed your available storage within a year
- If you exceed available storage, you will be able to move blob stores to larger storage devices.

You should separate repositories into two or more blob stores per storage device if both:

- You expect to exceed your currently available storage within a year
- You cannot move blob stores to larger storage devices, so you must add capacity to Nexus Repository Manager by adding additional storage devices.

The most flexible approach is to create a separate blob store for each repository, although this is not recommended except in extreme cases of unpredictable capacity because of the administrative complexity. The current repository-to-blob store limitations will be removed in an upcoming release of Nexus Repository Manager, which will make it possible to revise your repository/blob store approach over time.

Estimating Blob Store Size

Blob stores contain two files for each binary component stored in Nexus Repository Manager:

- The binary component, stored as a .bytes file (whose size is the same as the component)
- A properties file that stores a small amount of metadata for disaster recovery purposes (<1k)

The total storage size of a blob store is therefore approximately the total size of all of your components, plus an allowance for the properties files and the block size of your storage device. (On average, this will be $1.5 * \# \text{ of components} * \text{block size.}$)

Choosing the Blob Store Type

NXRM supports two blob store types:

- *File* blob store: stores the blobs on the filesystem. This is the default blob store type recommended for most installations.
- *S3* blob store: stores the blobs in an AWS S3 bucket. This is only recommended for Nexus Repository Manager installations deployed to AWS where storing data to S3 is preferable.

Setting up an S3 Bucket for a Blob Store

 Nexus Repository Manager only supports AWS S3. Other implementations of the S3 protocol may or may not work.

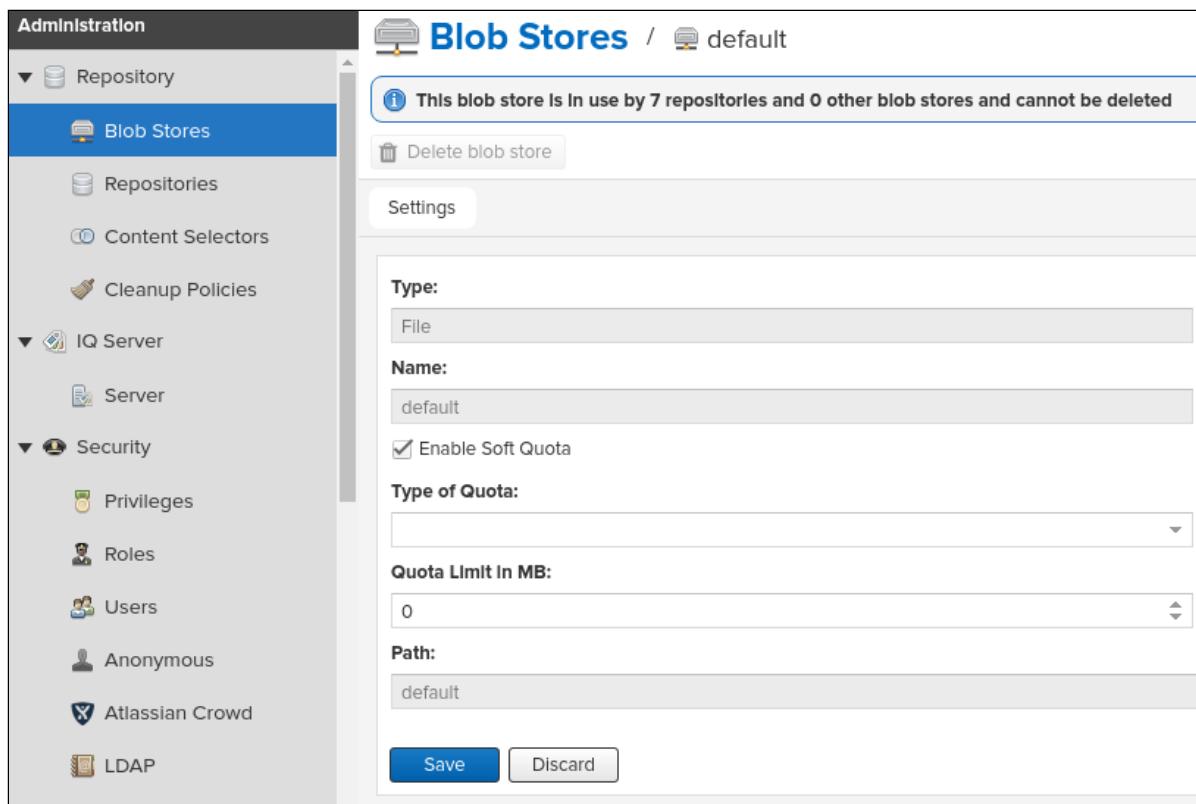
Nexus Repository Manager will automatically create an S3 bucket when a blob store is created, if it does not already exist. However, you may also create the bucket yourself. Note that:

- Nexus Repository Manager will automatically apply a lifecycle rule to expire deleted content.
- The bucket can use server-side encryption with KMS key management transparently. Other methods of server side encryption are not supported.
- If running on EC2 instances, Nexus Repository Manager can use the IAM role assigned to those instances when accessing S3 buckets.

Adding a Soft Quota

 A soft quota will never block read or write operations on the blob store

The configuration for a soft quota can be found in the *Blob Stores* feature view, located in the *Blob Stores* sub menu of the *Administration* menu. A soft quota is a feature that monitors a blob store and raises an alert when it exceeds a constraint. The initial view displayed in *Figure: "Example Soft Quota Configuration"* shows the parameters of a soft quota: *Type of Quota* and *Quota Limit in MB*.



This screenshot shows the 'Blob Stores' configuration page in the Nexus Repository Manager 3 interface. The left sidebar has the 'Administration' menu expanded, with 'Repository' selected. Under 'Repository', 'Blob Stores' is also selected, highlighted in blue. The main content area has a header 'Blob Stores / default'. A warning message at the top states: 'This blob store is in use by 7 repositories and 0 other blob stores and cannot be deleted'. Below this are several configuration fields:

- Type:** File
- Name:** default
- Enable Soft Quota:**
- Type of Quota:** (dropdown menu)
- Quota Limit in MB:** 0
- Path:** default

At the bottom are 'Save' and 'Discard' buttons.

Figure: Example Soft Quota Configuration

Space Used and *Space Remaining* are the two possible choices for *Type of Quota*. A *Space Used* quota alerts when the total bytes used by a blob store exceeds the *Quota Limit in MB*. *Space Remaining* quotas are triggered when the space available for new blobs drops below the *Quota Limit in MB*. When a soft quota is violated, that information will be available in the following locations:

- A *WARN* level message in the logs
- In the *Status* item in the *Support* sub menu of the *Administration* menu, the status of all soft quotas is aggregated in the *BlobStores* status
- A REST API endpoint

Proxy Repository

A repository with the type *proxy*, also known as a proxy repository, is a repository that is linked to a remote repository. Any request for a component is verified against the local content of the proxy repository. If no local component is found, the request is forwarded to the remote repository. The component is then retrieved and stored locally in the repository manager, which acts as a cache. Subsequent requests for the

same component are then fulfilled from the local storage, therefore eliminating the network bandwidth and time overhead of retrieving the component from the remote repository again.

By default, the repository manager ships with the following configured proxy repositories:

maven-central

This proxy repository accesses the [Central Repository](#)⁶⁸⁰, formerly known as Maven Central. It is the default component repository built into Apache Maven and is well-supported by other build tools like Gradle, SBT or Ant/Ivy.

nuget.org-proxy

This proxy repository accesses the [NuGet Gallery](#)⁶⁸¹. It is the default component repository used by the nuget package management tool used for .Net development.

Hosted Repository

A repository with the type *hosted*, also known as a hosted repository, is a repository that stores components in the repository manager as the authoritative location for these components.

By default, the repository manager ships with the following configured hosted repositories:

maven-releases

This hosted repository uses the maven2 repository format with a release version policy. It is intended to be the repository where your organization publishes internal releases. You can also use this repository for third-party components that are not available in external repositories and can therefore not be retrieved via a configured proxy repository. Examples of these components could be commercial, proprietary libraries such as an Oracle JDBC driver that may be referenced by your organization.

maven-snapshots

This hosted repository uses the maven2 repository format with a snapshot version policy. It is intended to be the repository where your organization publishes internal development versions, also known as snapshots.

nuget-hosted

This hosted repository is where your organization can publish internal releases in repository using the NuGet repository format. You can also use this repository for third-party components that are not available in external repositories, that could potentially be proxied to gain access to the components.

⁶⁸⁰ <http://search.maven.org/>

⁶⁸¹ <https://www.nuget.org/>

Repository Group

A repository with the type *group*, also known as repository group, represents a powerful feature of Nexus Repository Manager. They allow you to combine multiple repositories and other repository groups in a single repository. This in turn means that your users can rely on a single URL for their configuration needs, while the administrators can add more repositories and therefore components to the repository group.

The repository manager ships with the following groups:

maven-public

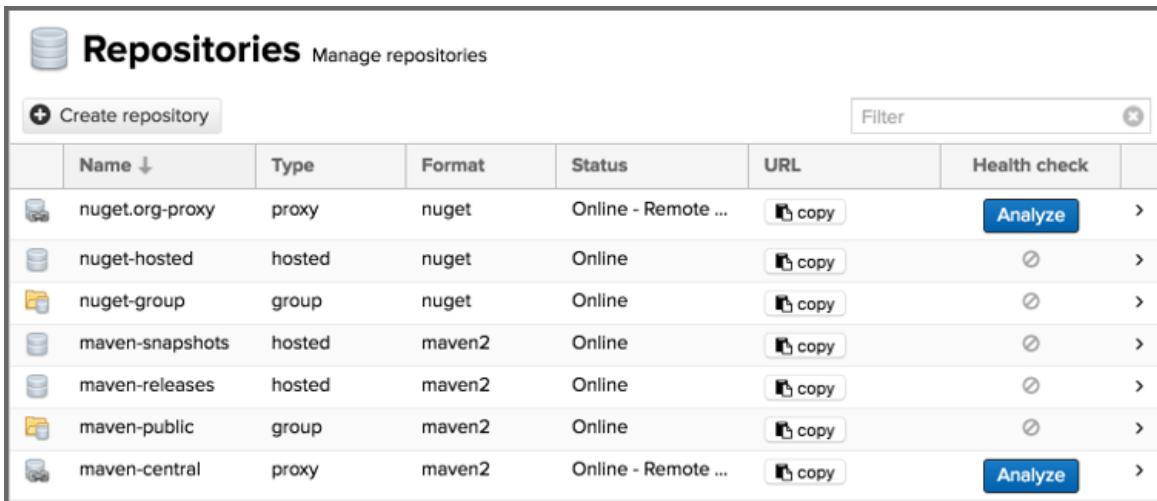
The maven-public group is a repository group of maven2 formatted repositories and combines the important external proxy repository for the Central Repository with the hosted repositories *maven-releases* and *maven-snapshots*. This allows you to expose the components of the Central Repository as well as your internal components in one single, simple-to-use repository and therefore URL.

nuget-group

This group combines the nuget formatted repositories *nuget-hosted* and *nuget.org-proxy* into a single repository for your .Net development with NuGet.

Managing Repositories and Repository Groups

The administration user interface for repositories and repository groups is available via the *Repositories* item in the *Repository* sub menu of the *Administration* menu. It allows you to create and configure repositories as well as delete them and perform various maintenance operations. The initial view displayed in *Figure: "List of Repositories"* features a list of all configured repositories and repository groups.



	Name ↓	Type	Format	Status	URL	Health check	
1	nuget.org-proxy	proxy	nuget	Online - Remote ...		Analyze	>
2	nuget-hosted	hosted	nuget	Online			>
3	nuget-group	group	nuget	Online			>
4	maven-snapshots	hosted	maven2	Online			>
5	maven-releases	hosted	maven2	Online			>
6	maven-public	group	maven2	Online			>
7	maven-central	proxy	maven2	Online - Remote ...		Analyze	>

Figure: List of Repositories

The list of repositories displays some information for each repository in the following columns:

Name

the unique name of the repository or repository group

Type

the type of the repository with values of *proxy* or *hosted* for repositories or group for a repository group

Format

the repository format used for the storage in the repository with values such as *maven2*, *nuget* or others

Status

the status of the repository as well as further information about the status. A functioning repository would show the status to be *Online*. Additional information can e.g., be about SSL certification problems or the status of the remote repository for a currently disabled proxy repository

URL

the *copy* button prompts a dialog containing a direct URL path exposing the repository

Health Check

displays the repository health statistics from a previously run [Repository Health Check](#)⁶⁸², or a button to start the analysis

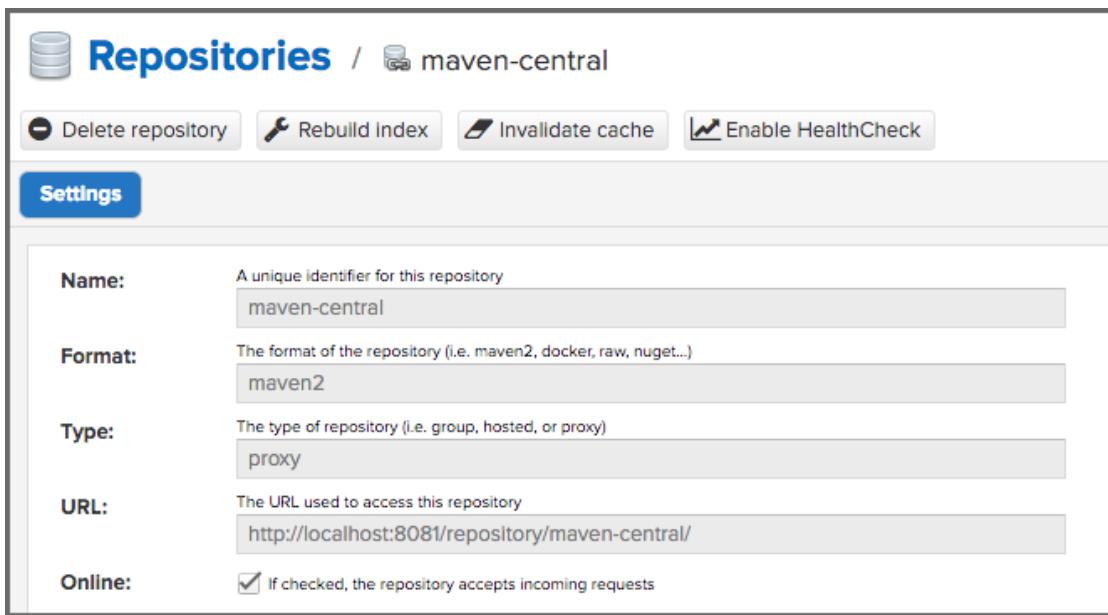
The *Create repository* button above the repository list triggers a dialog to select the Recipe for the new repository. The recipe combines the format and the type of repository into a single selection. Depending on your repository manager version and installed plugins, the list of available choices differs.

For example to create another release repository in *maven2* format, you would click on the row with the recipe *maven2 (hosted)* in the dialog. If you wanted to proxy a *maven2* repository, choose *maven 2 (proxy)*. On the other hand if you want to proxy a *nuget* repository, choose *nuget (proxy)*. With *maven2 (group)* you can create a repository group for *maven2* repositories.

After this selection, you are presented with the configuration view, that allows you to fill in the required parameters and some further configuration. The exact details on the view depend on the selected repository provider and are identical to the administration for updating the configuration of a repository documented in the following sections.

Once you have created a repository or repository group, it is available in the list for further configuration and management. Clicking on a specific row allows you to navigate to this repository specific administration section. An example for the maven-central repository is partially displayed in *Figure: "Partial Repository Configuration for a Proxy Repository"*.

⁶⁸² <http://blog.sonatype.com/how-to-use-the-new-repository-health-check-2.0>



The screenshot shows the 'Repositories' section of the Nexus Repository Manager 3 interface. A specific repository named 'maven-central' is selected. The configuration details are as follows:

- Name:** maven-central
- Format:** maven2
- Type:** proxy
- URL:** http://localhost:8081/repository/maven-central/
- Online:**

Figure: Partial Repository Configuration for a Proxy Repository

The repository administration feature view has buttons to perform various actions on a repository. The buttons displayed depend on the repository format and type. The following buttons can be found:

Delete repository

The *Delete repository* button allows you to delete the repository and all related configuration and components, after confirming the operation in a dialog.

Invalidate cache

The *Invalidate cache* button invalidates the caches for this repository. The exact behavior depends on the repository type:

Proxy repositories

Invalidating the cache on a proxy repository clears the proxy cache such that any items cached as available will be checked again for any changes the next time they are requested. This also clears the negative cache for the proxy repository such that any items that were not found within the defined cache period will be checked again the next time they are requested.

Repository groups

Invalidating the cache of a repository group, clears the group cache such that any items fetched and held in the group cache, such as Maven metadata, will be cleared. This action also invalidates the caches of any proxy and group repositories that are members of this group.

Rebuild Index

The *Rebuild Index* button allows you to drop and recreate the search index for the proxy repository, synchronizing the contents with search index. This button is only available for proxy repositories.

The following properties can be viewed for all repositories and can not be edited after the initial creation of the repository:

Name

The *Name* is the identifier that will be used in the URL for access to the repository. For example, the proxy repository for the Central Repository has a name of maven-central. The *Name* must be unique in a given repository manager installation and is required.

Format

Format defines in what format the repository manager exposes the repository to external tools. Supported formats depend on the edition of the repository manager and the installed plugins. Examples are *maven2*, *nuget*, *raw*, *docker*, *npm*.

Type

The type of repository - *proxy*, *hosted* or *group*.

URL

It shows the user facing URL this means that Maven and other tools can access the repository directly at e.g., `http://localhost:8081/repository/maven-central`.

Online

The checkbox allows you set whether this repository is available to client side tools or not.

Beyond the generic fields used for any repository, a number of different fields are used and vary depending on the repository format and type. They are grouped under a number of specific headers that include configuration for the related aspects and include:

- Storage
- Hosted
- Proxy
- Negative Cache
- HTTP
- Maven 2
- NuGet
- and others

Storage

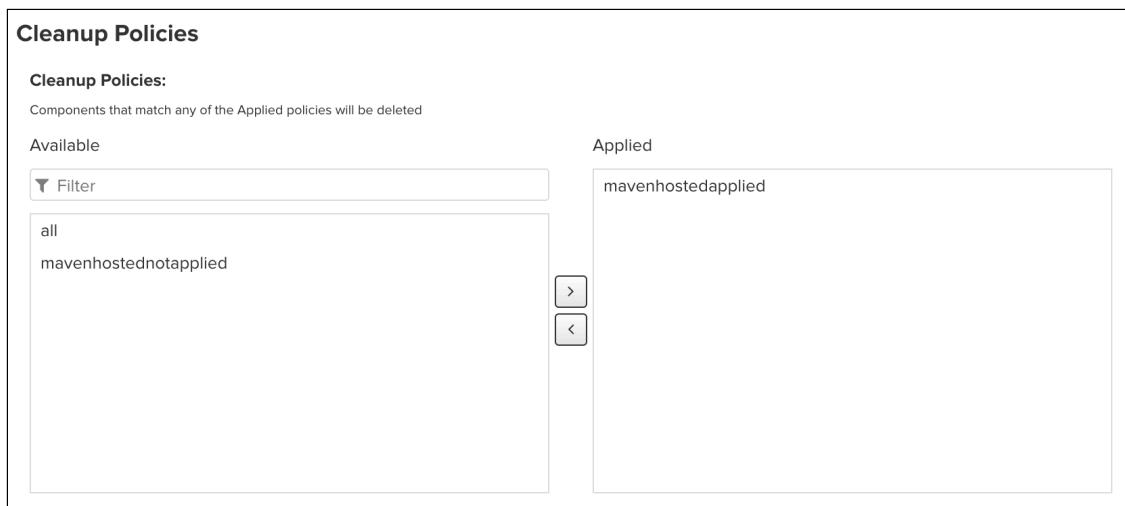
Every repository needs to have a Blob store configured to determine where components are stored. The drop-down allows you to select from all the configured blob stores. Documentation about creating blob stores can be found in [Blob Stores\(see page 235\)](#).

The *Strict Content Type Validation* allows you to activate a validation that checks the MIME type of all files published into a repository to conform to the allowed types for the specific repository format.

Cleanup Policies

This section, shown in *Figure: Cleanup Policies subsection*, provides a list of cleanup policies that can be used against hosted and proxy repositories. When the *Admin - Cleanup repositories using their associated policies* task is executed it pares the contents of that repository against all *Applied* policies by deleting contents that match any of them. Ultimately, this is a way of controlling your disk space and making sure unused items are removed from NXRM. The use of this area is optional and by default the *Applied* section is empty.

The lists will only include cleanup policies you have created for the respective format. See [here\(see page 297\)](#) for more about creating and maintaining cleanup policies.



Cleanup Policies

Cleanup Policies:
Components that match any of the Applied policies will be deleted

Available

Filter

all
mavenhostednotapplied

Applied

mavenhostedapplied

Figure: Cleanup Policies subsection

Groups do not use cleanup policies themselves, they simply contain the components from their children.

- ❗ It is important to note that any component affected by ANY policy in the *Applied* section will be deleted. So if you have a policy that deletes only X and a policy that deletes only Y, both X and Y will be removed when the task executes if both policies are *Applied*.

Hosted

A hosted repository includes configuration of a Deployment policy in the Hosted configuration section. Its setting controls how a hosted repository allows or disallows component deployment.

If the policy is set to *Read-only*, no deployment is allowed.

If this policy is set to *Disable redeploy*, a client can only deploy a particular component once and any attempt to deploy a component again will result in an error. The disabled redeploy is the default value, since most client tools assume components to be immutable and will not check a repository for changed components that have already been retrieved and cached locally.

If the policy is set to *Allow redeploy*, clients can deploy components to this repository and overwrite the same component in subsequent deployments.

Proxy

The configuration for proxy repositories in the Proxy section also contains the following parameters:

Remote Storage

A proxy repository on the other hand requires the configuration of the *Remote Storage*. It needs to be configured with the URL of the remote repository, that should to be proxied. When selecting the URL to proxy it is beneficial to avoid proxying remote repository groups. Proxying repository groups prevents some performance optimization in terms of accessing and retrieving the content of the remote repository. If you require components from the group that are found in different hosted repositories on the remote repository server it is better to create multiple proxy repositories that proxy the different hosted repositories from the remote server on your repository manager instead of simply proxying the group.

Use the Nexus truststore

This checkbox allows you to elect for the repository manager to manage the SSL certificate of the remote repository. It is only displayed - if the remote storage uses a HTTPS URL. The *View certificate* button triggers the display of the SSL certificate details in a dialog. The dialog allows you to add or remove the certificate from the certificate truststore maintained by the repository manager. Further details are documented in [Outbound SSL - Trusting SSL Certificates of Remote Repositories](#)(see page 339).

Blocked

Setting a repository to blocked causes the repository manager to no longer send outbound requests to the remote repository.

Auto blocking enabled

If Auto blocking enabled is set to true, the repository manager automatically blocks a proxy repository if the remote repository becomes unavailable. While a proxy repository is blocked, components will still be served to clients from a local cache, but the repository manager will not attempt to locate a component in a remote repository. The repository manager periodically retests the remote repository and unblocks it once it becomes available.

Maximum component age

When the proxy receives a request for a component, it does not request a new version from the remote repository until the existing component is older than Maximum component age .

Maximum metadata age

The repository manager retrieves metadata from the remote repository. It will only retrieve updates to metadata after the Maximum metadata age has been exceeded. If the metadata is component metadata, it uses the longer of this value and Maximum component age before rechecking.

Negative Cache

Not found cache enabled/Not found cache TTL

If the repository manager fails to locate a component, it will cache this result for a given number of minutes. In other words, if the repository manager can't find a component in a remote repository, it will not perform repeated attempts to resolve this component until the *Not found cache TTL* time has been exceeded. The default for this setting is 1440 minutes (or 24 hours) and this cache is enabled by default.

HTTP

The HTTP configuration section allows you to configure the necessary details to access the remote repository, even if you have to provide authentication details in order to access it successfully or if you have to connect to it via a proxy server.

 This configuration is only necessary, if it is specific to this repository. Global HTTP proxy and authentication is documented in [HTTP and HTTPS Request and Proxy Settings](#)(see page 266).

Authentication

This section allows you to select *Username* or *Windows NTLM* as *Authentication type*. Subsequently you can provide the required *Username* and *Password* for plain authentication or *Username*, *Password*, *Windows NTLM hostname* and *Windows NTLM domain* for *Windows NTLM*-based authentication.

HTTP request settings

In the HTTP request settings you can change properties of the HTTP requests to the remote repository. The values you can apply to this section are:

User-agent customization

Enter the string to be appended to user-agent HTTP headers.

Connection retries

Enter the total number of connection attempts after an initial timeout.

Connection timeout

Set the timeout interval for requests, in seconds.

Enable circular redirects

Allow proxy repositories to follow redirects indicated by the remote server even if they point to an already processed URL.

Enable cookies

Authorize HTTP cookies sent by the remote server, for future requests.

ⓘ <https://maven.oracle.com>⁶⁸³ is a server that requires both *Enable circular redirects* and *Enable cookies*. This is because, when requesting data you are redirected to a queue of different URLs, most of which are involved with authentication.

By enabling these options, you allow the repository manager to maintain the authentication state in a cookie that would be sent with each request, eliminating the need for the authentication-related redirects and avoiding timeouts.

Changes made to *HTTP request settings* are applied to all HTTP requests made from the repository manager to the remote repository being proxied. Enabling these settings will override any general settings defined in [HTTP and HTTPS Request and Proxy Settings](#)(see page 266).

Some repository formats include configuration options, such as these formats:

- *Repository Connectors*, *Docker Registry API Support*, and (for proxies) *Docker Index* for Docker repositories
 - [SSL and Repository Connector Configuration](#)(see page 384)
 - [Support for Docker Registry API](#)(see page 386)
 - [Proxy Repository for Docker](#)(see page 386)
- *Maven 2* for Maven repositories: [Maven Repository Format](#)(see page 412)
- *NuGet* for NuGet proxy repositories: [NuGet Repository Format](#)(see page 428)
- *Bower* for Bower proxy repositories: [Proxying Bower Repositories](#)(see page 372)
- *Yum* for Yum hosted repositories: [Hosting Yum Repositories](#)(see page 461)

Repository Groups

The creation and configuration for a repository group differs a little from pure repositories. It allows you to manage the member repositories of a repository group. An example for a repository group using the *maven2* format is visible in *Figure: “Repository Group Configuration”*. In this figure you can see the contents of the *maven-public* group that is pre-configured in Nexus Repository Manager.

683 <https://maven.oracle.com/>

Repositories / maven-public

[Delete repository](#) [Invalidate cache](#)

Settings

Name: A unique identifier for this repository
maven-public

Format: The format of the repository (i.e. maven2, docker, raw, nuget...)
maven2

Type: The type of repository (i.e. group, hosted, or proxy)
group

URL: The URL used to access this repository
http://localhost:8081/repository/maven-public/

Online: If checked, the repository accepts incoming requests

Storage

Blob store:
Blob store used to store asset contents
default

Strict Content Type Validation:
 Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

Group

Member repositories:
Select and order the repositories that are part of this group

Available	Members
<input type="text" value="Filter"/> x	^ > < v
	maven-releases maven-snapshots maven-central

Save **Discard**

Figure: Repository Group Configuration

The *Format* and *Type* are determined by the selection of the provider in the creation dialog e.g., *maven2 (group)* for the *maven-public* as a *maven2* format repository group.

The *Name* is set during the creation and is fixed once the repository group is created.

The *Online* checkbox allows you set whether this repository group is available to client side tools or not.

The *Member repositories* selector allows you to add repositories to the repository group as well as remove them. The *Members* column includes all the repositories that constitute the group. The *Available* column includes all the repositories and repository groups that can potentially be added to the group.

Note that the order of the repositories listed in the *Member* section is important. When the repository manager searches for a component in a repository group, it will return the first match. To reorder a repository in this list, click and drag the repositories and groups in the *Members* list or use the arrow buttons between the *Available* and *Members* list. These arrows can be used to add and remove repositories as well.

The order of repositories or other groups in a group can be used to influence the effective metadata that will be retrieved by Maven or other tools from a repository group. It is recommended practice to place hosted repositories higher in the list than proxy repositories. For proxy repositories, the repository manager needs to check the remote repository which will incur more overhead than a hosted repository lookup.

It is also recommended to place repositories with a higher probability of matching the majority of components higher in this list. If most of your components are going to be retrieved from the Central Repository, putting *maven-central* higher in this list than a smaller, more focused repository is going to be better for performance, as the repository manager is not going to interrogate the smaller remote repository for as many missing components. These best practices are implemented in the default configuration.

Repository Management Example

The following sections detail some common steps of your repository management efforts on the example of a *maven2* repository.

Adding Repositories for Missing Dependencies

If you've configured your Maven settings.xml or other build tool configuration to use the *maven-public* repository group as a mirror for all repositories, you might encounter projects that are unable to retrieve components from your local repository manager installation.

- ⓘ More details about client tool configuration for Maven repositories can be found in [Maven Repositories](#)(see page 411).

This usually happens because you are trying to build a project that has defined a custom set of repositories and snapshot repositories or relies on the content of other publicly available repositories in its configuration. When you encounter such a project all you have to do is add this repository as a new *maven2* format, proxy repository and then add the new proxy repository to the *maven-public* group.

The advantage of this approach is that no configuration change on the build tool side is necessary at all.

Adding a New Repository

Once you have established the URL and format of the remote repository you are ready to configure the repository. E.g. the [JBoss.org](http://jboss.org)⁶⁸⁴ releases repository contains your missing component. Click on the *Create repository* button in the *Repositories* feature view and click on *maven2 (proxy)* from the list in the dialog.

In the configuration dialog:

- Set *Name* to *jboss-releases*
- Set *Remote storage* to <https://repository.jboss.org/nexus/content/repositories/releases/>
- For a *maven2* format repository, confirm that the *Version policy* is set correctly to *Release*.
- Click on the *Create repository* button at the end of the form

The repository manager is now configured to proxy the repository. If the remote repository contains snapshots as well as release components, you will need to repeat the process creating a second proxy repository with the same URL setting version policy to *Snapshot*.

Adding a Repository to a Group

Next you will need to add the new repository *jboss-releases* to the *maven-public* repository group. To do this, click on the row of the *maven-public* group in the *Repositories* feature view.

To add the new repository to the public group, find the repository in the *Available* list on the left, click on the repository you want to add and drag it to the right to the *Members* list. Once the repository is in that list, you can click and drag the repository within that list to alter the order in which the group will be searched for a matching component. Press the *Save* button to complete this configuration.

In the last few sections, you learned how to add new repositories to a build in order to download components that are not available in the Central Repository.

If you were not using a repository manager, you would have added these repositories to the *repository* element of your project's POM, or you would have asked all of your developers to modify `~/.m2/settings.xml` to reference two new repositories. Instead, you used the repository manager to add the two repositories to the public group. If all of the developers are configured to point to the public group, you can freely swap in new repositories without asking your developers to change local configuration, and you've gained a certain amount of control over which repositories are made available to your development team. In

⁶⁸⁴ <http://jboss.org/>

addition the performance of the component resolving across multiple repositories will be handled by the repository manager and therefore be much faster than client side resolution done by Maven each time.

Content Selectors

Content selectors provide a means for you to select specific content from all of your content. The content you select is evaluated against expressions written in CSEL (Content Selector Expression Language). CSEL is a light version of JEXL used to script queries along specific paths and coordinates available to your repository manager formats.

What Happened to JEXL?

We discovered that JEXL based content selectors weren't meeting performance expectations. To address this we introduced CSEL based selectors to support changes coming in future releases.

When you upgrade, Nexus Repository will automatically upgrade as many of your existing JEXL selectors to CSEL selectors as possible. Any remaining JEXL selectors will continue to function, but we encourage you to perform an upgrade as soon as possible.

Content selectors allow you to define what content users are allowed to access. You can define, in a simplified example, a selector named "Apache Maven" with a search expression of path =~ "^/org/apache/maven/". This would match all components that start with the designated component path.

Creating a Query

Before you identify user permissions for your selector, create the query first. Click *Content Selectors* located in *Repository*, from the *Administration* menu. Click *Create selector* to open a new form.

In the *Selector ID* section enter a *Name* and (optional) *Description* of your selector in the corresponding fields. In the *Specification* section use the *Search expression* field to build your query using CSEL syntax.

You can preview your selector and what results it will return by clicking the *Preview results* button. On click, a modal will appear as shown in *Figure: "Content Selector Preview Modal"*. The *Expression* field will automatically be filled in with anything you had in the *Search expression* field. Similarly, any changes to *Expression* will be saved to the *Search expression* when you close the preview.

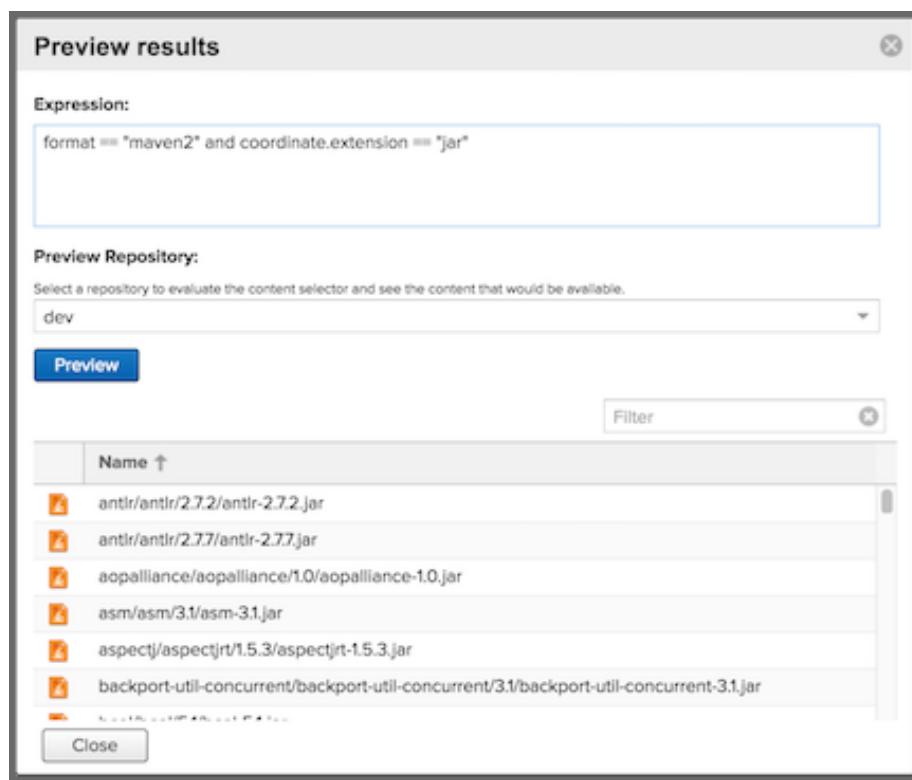


Figure: Content Selector Preview Modal

To see results your selector would find, select a repository or grouping of repositories from the *Preview Repository* dropdown and click the *Preview* button. Assets that match will be returned in the space below the filter and can be filtered upon if you wish to check on a specific result. The Name column is also sortable in ascending or descending order. Close returns you to the content selector creation screen.

Once you are satisfied with your fields, click *Create selector* to create the Content Selector. All saved selector queries you create will be listed in the *Content Selectors* screen.

Managing Selector Permissions

As part of your security setup, you can create user permissions to manage the filters you built in the *Create Selector* form. You can add a new privilege that controls operations such as *read*, *edit*, *delete*, and *** (all), for components matching that selector. The privilege can even span multiple repositories.

To create a new content selector privilege, click *Privileges* in the *Security* section of the Administration panel. Then click the *Create Privilege* button. Locate and click *Repository Content Selector* from the list of options in *Select Privilege Type*. You will see a form that displays the following:

- *Name*: Create a name for the content selector privilege.
- *Description*: Add a brief description for the privilege.
- *Content Selector*: Use this dropdown to select from a list of selectors you created.
- *Repository*: Use this dropdown to select from either a range of all repository contents, all repository contents of an individual format, or repositories created by you.

- **Actions:** Grant *read*, *edit*, *delete*, or * (all) privileges for user access control.

To complete the form, save the new privilege by clicking *Create privilege*. You can use your new privilege to regulate what permissible data you want the user to access. You could group all related privileges into a role as documented in [Roles\(see page 317\)](#). Ultimately, you could assign your roles to a user, as mentioned in [Users\(see page 320\)](#).

A practical example might be where you delegate all control of components in org.apache.maven to a "Maven" team. This way, you would not need to create separate repositories for each logical division of your components.

Content Selector Reference

Below are the allowable attributes for content selectors that define *path* and *format* as values supported by Nexus Repository Manager.

Attribute	Allowed Values
path	The path of your repository content
format	The format of the content for which you query

Content Selector Examples

Valid operators:

- **==**
Matches text exactly.
e.g. `format == "raw"`
- **=~**
Matches a Java regular expression pattern.
e.g. `path =~ "^/org/apache/commons/.+"`
- **=^**
Starts with text.
e.g. `path =^ "/com/example/"`
- **and**
Match all expressions.
e.g. `format == "maven2" and path =~ "^/org/apache/commons/.+"`

- **or**

Match any expression.

e.g. `format == "maven2" or format == "npm"`

- **(expr)**

Group multiple expressions.

e.g. `format == "npm" or (format == "maven2" and path =~ "^/org/apache/commons/.*)"`

Usage examples:

Select all raw format content

`format == "raw"`

Select all maven2 content along a path that starts with `org.apache.commons`

`format == "maven2" and path =~ "^/org/apache/commons/.+"`

i When writing a content selector, remember that the asset's path will always begin with a leading slash when the selector is evaluated. This is true even though the leading slash is not displayed when searching or browsing assets.

i Remember that to allow proper access when using Tree browsing in the UI, the content selectors need to include permissions for parent directories of the artifacts. Following along with the apache-commons examples used above, you would want something similar to this: `format == "maven2" and path =~ "/|/org/|/org/apache/|/org/apache/commons/.*"`. Alternatively, if you don't mind users being able to see any directory name (just not the contents), you could use `format== "maven2" and path =~ ".*/|/org/apache/commons.*"`.

Support Features

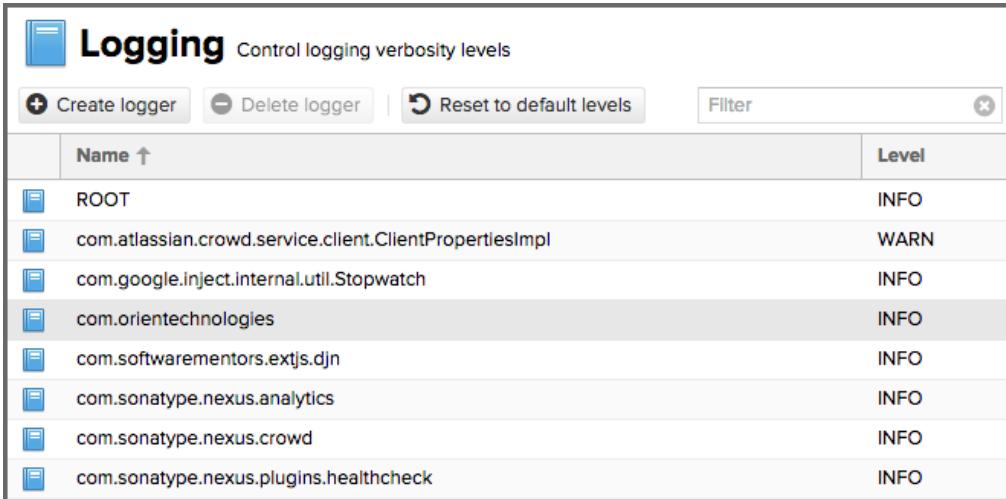
i Available in Nexus Repository OSS and Nexus Repository Pro

Nexus Repository Manager provides a number of features that allow you to ensure your server is configured correctly and provides you with tools to investigate details about the configuration. This information can be useful for troubleshooting and support activities.

All support features are available in the *Support* group of the *Administration* menu in the main menu section.

Logging and Log Viewer

You can configure the level of logging for the repository manager and all plugins as well as inspect the current log using the user interface with the *Logging* and the *Log Viewer* feature views. Access the Logging feature view displayed in *Figure: “The Logging Feature View for Configuring Loggers”* with the *Logging* menu item in the *Support* section in the *Administration* main menu.



Name ↑	Level
ROOT	INFO
com.atlassian.crowd.service.client.ClientPropertiesImpl	WARN
com.google.inject.internal.util Stopwatch	INFO
com.orientechnologies	INFO
com.softwarementors.extjs.djn	INFO
com.sonatype.nexus.analytics	INFO
com.sonatype.nexus.crowd	INFO
com.sonatype.nexus.plugins.healthcheck	INFO

Figure: The Logging Feature View for Configuring Loggers

The *Logging* feature view allows you to configure the preconfigured loggers as well as add and remove loggers. You can modify the log level for a configured logger by clicking on the *Level* value, e.g. `INFO`. It will change into a drop-down of the valid levels including `OFF`, `DEFAULT`, `INFO` and others. Press the *Update* button to apply the change.

The *Create logger* button can be used to create new loggers. You will need to know the Logger name you want to configure. Typically this corresponds to the Java package name used in the source code. Depending on your needs you can inspect the source of Nexus Repository Manager OSS and the plugins as well as the source of your own plugins to determine the related loggers or contact Sonatype support for detailed help.

If you select a row in the list of loggers, you can delete the highlighted logger by pressing the *Delete logger* button above the list. This only applies to previously created custom loggers. To disable a default configured logger, set it to `OFF`.

⚠ When upgrading the repository manager, keep in mind that some loggers change between versions, so if you rely on specific loggers, you might have to reconfigure them.

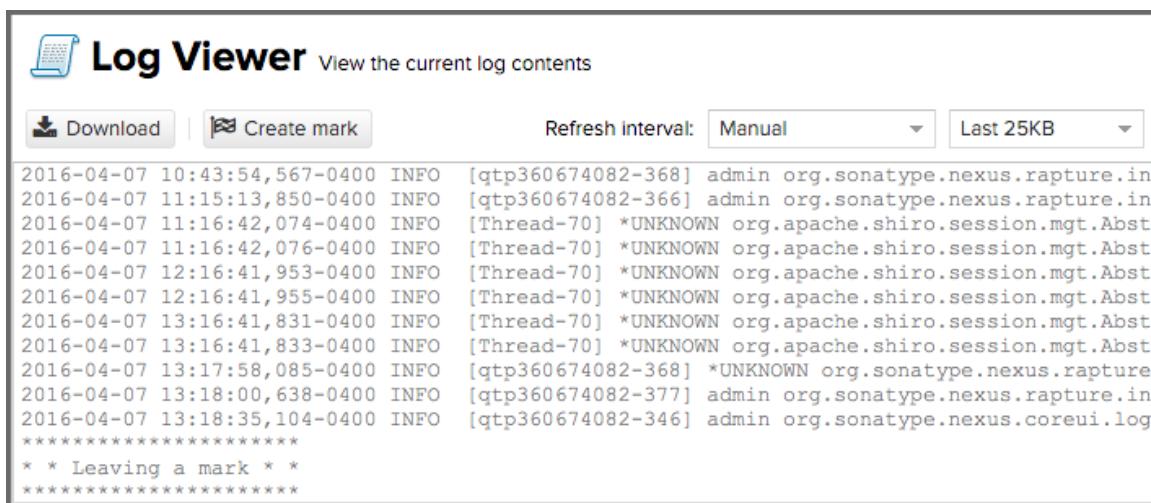
The *Reset to default levels* button allows you to remove all your custom loggers and get back to the setup shipped with a fresh install of the repository manager.

The loggers configured in the user interface are persisted into \$data-dir/etc/logback/logback-overrides.xml and override any logging levels configured in the main \$install-dir/etc/logback/logback.xml file as well as other logback-* files. If you need to edit a logging level in those files, edit the overrides file. This will give you access to edit the configuration in the user interface at a later stage and also ensure that the values you configure take precedence.

The ROOT logger level controls how verbose the logging is in general. If set to DEBUG, logging will be very verbose, printing all log messages including debugging statements. If set to ERROR, logging will be far less verbose, only printing out a log statement if the repository manager encounters an error. INFO represents an intermediate amount of logging.

When configuring logging, keep in mind that heavy logging can have a significant performance impact on an application and any changes trigger the change to the logging immediately.

Once logging is configured as desired, you can inspect the impact of your configuration in the *Log Viewer* feature view. It allows you to copy the log from the server to your machine by pressing the *Download* button. The *Create mark* button allows you to add a custom text string into the log, so that you can create a reference point in the log file for an analysis of the file. It will insert the text you entered surrounded by * symbols as visible in *Figure: "Viewing the Log with an Inserted Mark"*.



Log Viewer View the current log contents

Refresh interval:

```
2016-04-07 10:43:54,567-0400 INFO [qtp360674082-368] admin org.sonatype.nexus.rapture.in
2016-04-07 11:15:13,850-0400 INFO [qtp360674082-366] admin org.sonatype.nexus.rapture.in
2016-04-07 11:16:42,074-0400 INFO [Thread-70] *UNKNOWN org.apache.shiro.session.mgt.Abst
2016-04-07 11:16:42,076-0400 INFO [Thread-70] *UNKNOWN org.apache.shiro.session.mgt.Abst
2016-04-07 12:16:41,953-0400 INFO [Thread-70] *UNKNOWN org.apache.shiro.session.mgt.Abst
2016-04-07 12:16:41,955-0400 INFO [Thread-70] *UNKNOWN org.apache.shiro.session.mgt.Abst
2016-04-07 13:16:41,831-0400 INFO [Thread-70] *UNKNOWN org.apache.shiro.session.mgt.Abst
2016-04-07 13:16:41,833-0400 INFO [Thread-70] *UNKNOWN org.apache.shiro.session.mgt.Abst
2016-04-07 13:17:58,085-0400 INFO [qtp360674082-368] *UNKNOWN org.sonatype.nexus.rapture
2016-04-07 13:18:00,638-0400 INFO [qtp360674082-377] admin org.sonatype.nexus.rapture.in
2016-04-07 13:18:35,104-0400 INFO [qtp360674082-346] admin org.sonatype.nexus.coreui.log
*****
* * Leaving a mark * *
*****
```

Figure: Viewing the Log with an Inserted Mark

The *Refresh interval* configuration on the right on the top of the view allows you to configure the timing for the refresh as well as the size of the log displayed. A manual refresh can be triggered with the general refresh button in the main toolbar.

Metrics

The *Metrics* feature view is available in the *Support* section of the *Administration* main menu. It provides insight to characteristics of the Java virtual machine JVM running the repository manager and is displayed in *Figure: "JVM Metrics"*.

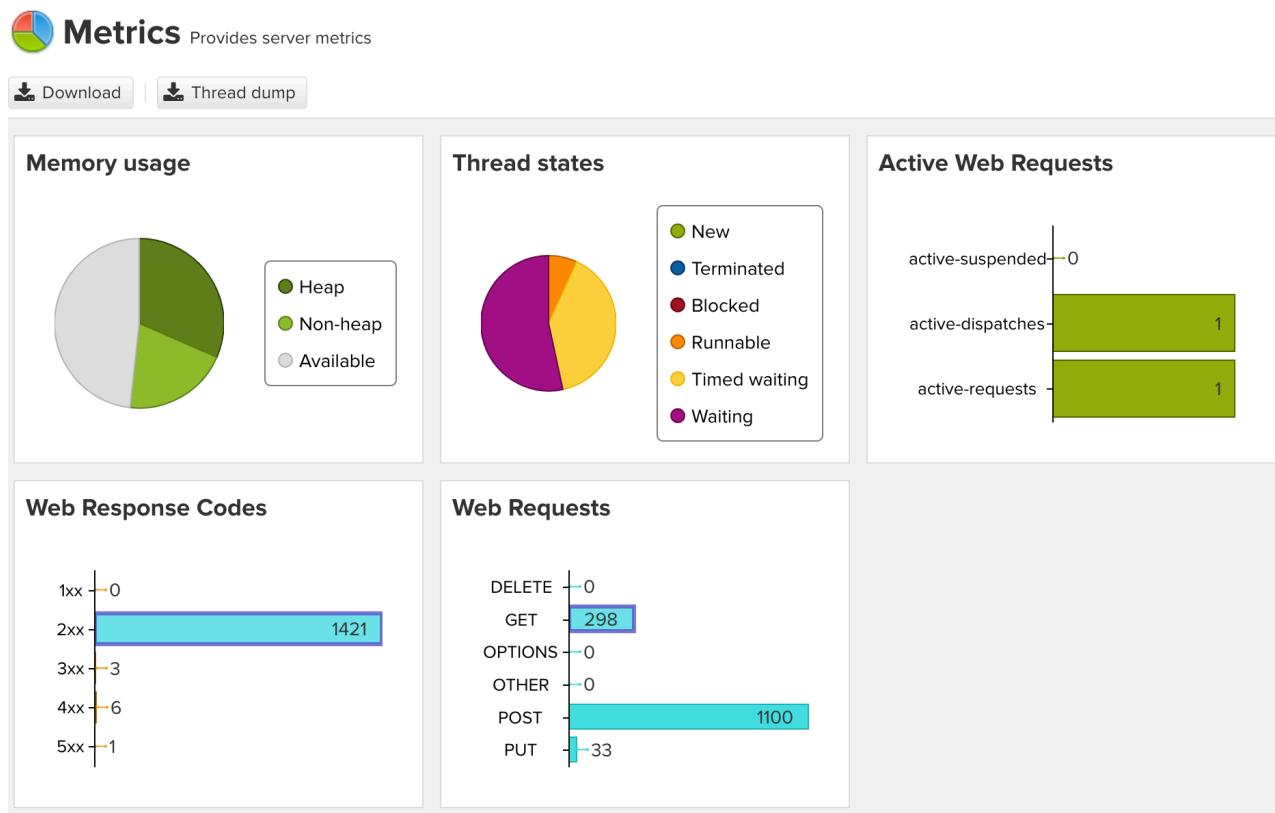


Figure: JVM Metrics

The *Memory usage*, *Thread states*, *Active Web Requests*, *Web Response Codes* and *Web Requests* charts provide some simple visualizations. Hovering over parts of the pie chart or bar graph provides a little more detail (e.g. a percentage of usage).

The *Download* button allows you to retrieve a large number of properties from the JVM and download them in a JSON-formatted text file. The *Thread dump* button triggers the creation of a thread dump of the JVM and a download of the resulting text file.

NEW IN 3.19 NXRM exports its metrics in Prometheus format at `/service/metrics/prometheus`.

Support ZIP

The *Support ZIP* feature view allows you to create a ZIP archive file that you can submit to Sonatype support via email or a support ticket. The checkboxes in *Contents* and *Options* allow you to control the content and size of the archive.

The repository manager implements security measures when a support ZIP is generated. Sensitive password related information is removed from generated files. When a support ZIP download is attempted, you may be prompted to verify your repository manager account credentials.

Support ZIP archive files are stored on the server under the `$data-dir/downloads` directory using a name which includes the time the file was generated.

Creating a Support ZIP

1. Sign in to the user interface using the default admin account or any account with the *nx-admin* role.
2. Click the cog icon  in the top toolbar to open the administration interface.
3. Select *Support* and then *Support ZIP* sidebar menu items.
4. Review the options and click the *Create Support ZIP* button. A popup dialog will be shown when the support zip generation is complete.
5. Either download the support ZIP using the *Download* button or use the file path shown to retrieve the file from the \$data-dir/downloads directory.

System Information

The *System Information* feature view displays a large number of configuration details related to:

Nexus Repository Manager

details about the versions of the repository manager and the installed plugins, install and work directory location, application host and port and a number of other properties

Java Virtual Machine

all system properties known by the JVM running the repository manager such as `java.runtime.name` and `os.name`

Operating System

including environment variables like `JAVA_HOME` or `PATH` as well as details about the runtime in terms of processor, memory and threads, network connectors and storage file stores

You can copy a subsection of the text from the panel or use the *Download* button to retrieve a JSON-formatted text file.

System Configuration

 Available in Nexus Repository OSS and Nexus Repository Pro

The *System* section of the Administration menu gives you access to a number of configuration features that you typically need to configure, after successful installation.

Accessing Information About Bundles

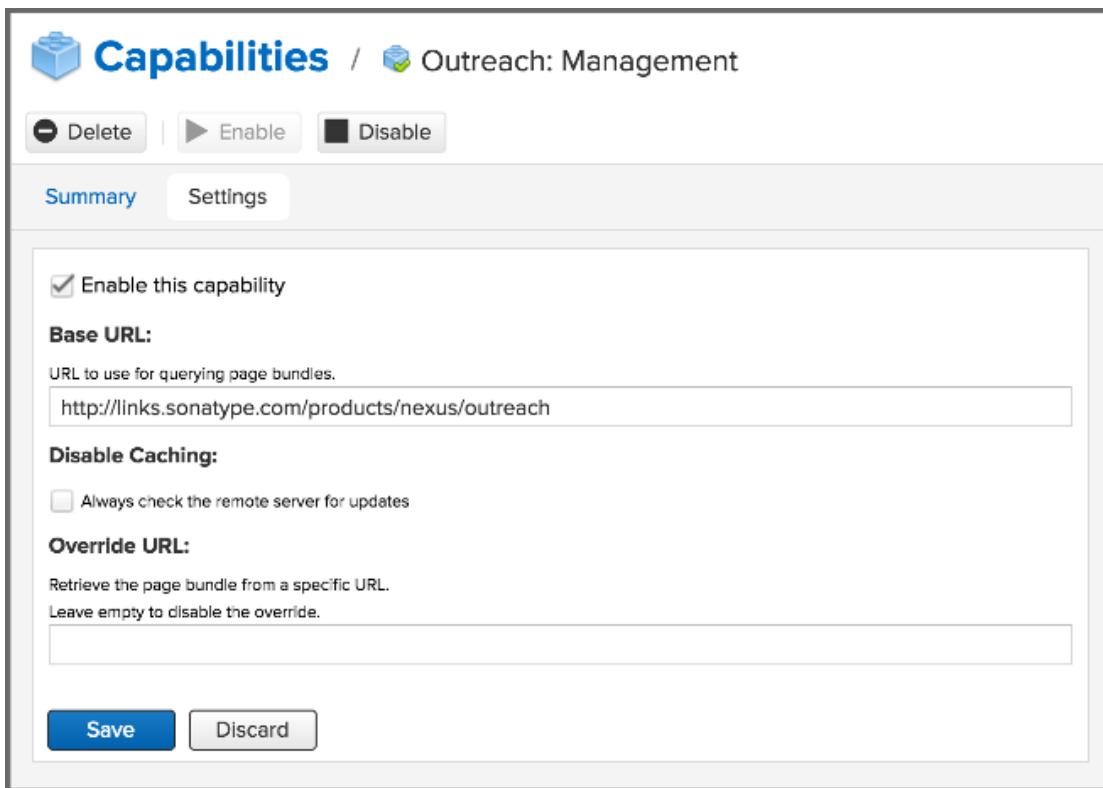
The Nexus Repository Manager application runs on the OSGi container Apache Felix. All features and plugins are managed by the container and are implemented as OSGi bundles. The *Bundles* feature view is available in the *System* section of the *Administration* main menu. It allows you to inspect a list of all the OSGi bundles that are deployed as part of the application and access detailed information about each bundle.

Find out more about OSGi and OSGi bundles on the website of the [OSGi Alliance](https://www.osgi.org/)⁶⁸⁵.

Accessing and Configuring Capabilities

Capabilities are features of the repository manager and plugins that can be configured by a user in a generic administration feature view accessible in the *System* section of the *Administration* main menu via *Capabilities*.

The repository manager ships with a number of capabilities preinstalled and allows you to enable/disable them. An example capability is *Outreach: Management* displayed in Figure: "Outreach: Management' Capability Settings".



The screenshot shows the 'Capabilities' feature view in the Nexus Repository Manager. The title bar says 'Capabilities / Outreach: Management'. There are three buttons at the top: 'Delete', 'Enable', and 'Disable'. Below the buttons are two tabs: 'Summary' (which is selected) and 'Settings'. The 'Summary' tab contains the following configuration:

- Enable this capability:** A checked checkbox.
- Base URL:** A label followed by a text input field containing 'http://links.sonatype.com/products/nexus/outreach'.
- Disable Caching:** A checkbox labeled 'Always check the remote server for updates' which is unchecked.
- Override URL:** A label followed by a text input field containing an empty string.

At the bottom of the 'Summary' tab are two buttons: 'Save' (in blue) and 'Discard'.

Figure: 'Outreach: Management' Capability Settings

⁶⁸⁵ <https://www.osgi.org/>

The list of existing capabilities can be filtered with the search input box in the header of the list and sorted by the different columns by pressing a column header. The list uses the following columns:

State

The State column does not have a title. Enabled capabilities have a green checkmark added on top of a blue icon. Disabled capabilities use a greyed out icon.

Type

The Type column provides the specific type of a capability in the list.

Category

The Category is optional and details the wider context the capability belongs to.

Description

The Description column contains further descriptive information about the capability.

Notes

The Notes column can contain user created text about the capability.

Every existing capability can be inspected and configured by selecting it in the list. The resulting view displays the *Summary* view of the specific capability. This view includes the display of Type, State and, optionally, Category and Description in the *Summary* section as well as further information in the Status, About, and Notes sections. The Status section displays a text message that details the status of the capability and any potential problems with the configuration. Depending on the capability, the reasons can vary widely. The About section displays a descriptive text about the purpose of the capability.

The Notes field can be used to provide a descriptive text about the capability or any other notes related to it and can be persisted by pressing the Save button.

The Delete button below the title allows you to remove a capability and its configuration entirely. The Enable and Disable buttons on the other hand can be used to switch the state of the capability. The Settings view allows you to activate or deactivate the capability with the *Enable this capability* checkbox. Below this checkbox, each capability type has specific additional configuration parameters available. Once you have completed the configuration, press the Save button.

The capabilities management feature view supports adding new capabilities by pressing the *Create capability* button above the list and selecting the desired capability Type from the list. The next view allows you to perform any capability-specific configuration and finish the process by pressing the *Create capability* button below the parameters.

Many of the built-in capabilities and plugins can be configured in the Capabilities administration section but also in other more user friendly, targeted user interface sections, e.g., the user token feature of Nexus Repository Manager Pro can be administrated by using the interface available via the *User Token* menu item in the Security section of the Administration menu as well as by editing the user token capability. Other capabilities are internal functionality and sometimes managed automatically by the responsible plugin. Some

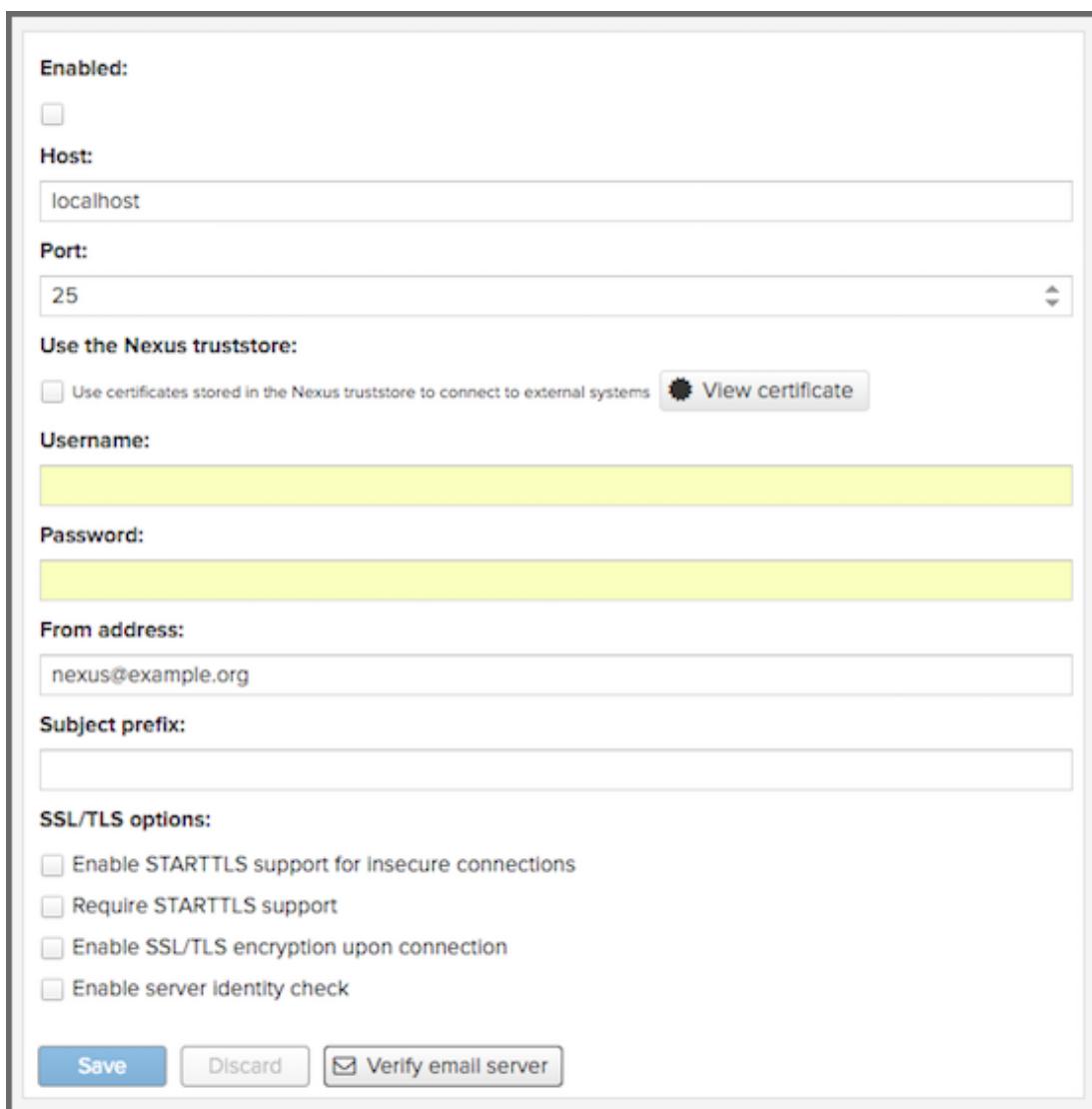
optional configuration like the branding plugin or advanced settings for the UI are only done in the capabilities administration.

Usage of specific capabilities to achieve a variety of tasks is detailed in parts of the documentation.

In many cases you will not need to configure anything in *Capabilities* unless explicitly instructed to do so by the support team. Execute any capability changes with caution, potentially backing up your configuration before proceeding.

Email/SMTP Server Configuration

The repository manager may send out email messages for a number of reasons. In order for these messages to be delivered, you need to configure the connection to the SMTP server under the *Email Server* menu item in the *System* section of the *Administration* menu as displayed in *Figure: "Email Server Configuration"*.



The screenshot shows the 'Email Server Configuration' dialog box. It contains the following fields:

- Enabled:** An unchecked checkbox.
- Host:** A text input field containing 'localhost'.
- Port:** A dropdown menu showing '25'.
- Use the Nexus truststore:** A section with a checked checkbox 'Use certificates stored in the Nexus truststore to connect to external systems' and a 'View certificate' button.
- Username:** A yellowed-out text input field.
- Password:** A yellowed-out text input field.
- From address:** A text input field containing 'nexus@example.org'.
- Subject prefix:** An empty text input field.
- SSL/TLS options:** A section with four unchecked checkboxes:
 - Enable STARTTLS support for insecure connections
 - Require STARTTLS support
 - Enable SSL/TLS encryption upon connection
 - Enable server identity check
- Action buttons:** 'Save' (blue), 'Discard' (grey), and 'Verify email server' (button with envelope icon).

Figure: Email Server Configuration

The following configuration options are available:

Enabled

Determines whether email sending is activated or not, independent of a server being configured.

Host and Port

The name of the host and the port to use to connect to the SMTP server.

Use the Nexus truststore

This checkbox allows you to configure the repository manager to use its certificate truststore. You can also view and import the certificate from the email server. Further details are documented in [Configuring SSL](#)(see page 338).

Username and Password

The credentials of the user of the SMTP server to use for authentication.

From address

This parameter defines the email address used in the "From:" header of any email sent by the repository manager. Typically, this is configured as a "Do-Not-Reply" email address or a mailbox or mailing list monitored by the administrators of the repository manager.

Subject prefix

This parameter allows you to define a prefix used in the subject line of all emails sent by the repository manager. This allows the recipients to set up automatic filtering and sorting easily. An example is [Nexus Notification].

SSL/TLS options

These options can be used to configure usage of Transport Layer Security (TLS) and Secure Sockets Layer (SSL) when emails are sent by the repository manager using SMTP. These options include the ability to use STARTTLS, which upgrades the initially established, plain connection to be encrypted when sending emails. The following options are available to you:

Enable STARTTLS support for insecure connections

This checkbox allows you to enable support for upgrading insecure connections using STARTTLS.

Require STARTTLS support

This checkbox requires that insecure connections are upgraded using STARTTLS. If this is not supported by the SMTP server, no emails are sent.

Enable SSL/TLS encryption upon connection

This checkbox enables SSL/TLS encryption for the transport on connection using SMTPS/POPS.

Enable server identity check

This option verifies the server certificate when using TLS or SSL, following the RFC 2595 specification.

Once you have configured the parameters you can use the *Verify email* server button to confirm the configured parameters and the successful connection to the server. You are asked to provide an email address that should receive a test email message. Successful sending is confirmed in a message.

Nodes

The Nodes screen provides a summary of all active nodes. The screen keeps a record of all running nodes that you can manage for single or multiple deployments of nodes, in table form. To view the status of your active node click *Nodes* in the *Administration* menu under *System*. When you click a row, you get a detailed summary of the chosen node.

The summary lists:

- *Node Identity*, a unique ID accessible via the System Information
- *Socket Address*, the address corresponding to the server

If you run multiple nodes the *Nodes* screen displays all synchronized nodes in the table. View the table to monitor and verify server connections.

Also, from the *Nodes* screen you can protect your server's database from write access by activating read-only mode. This allows you to avoid modifications to your server configuration and blob stores when performing system maintenance. To enable it:

1. Click *Nodes*, under *System* in the *Administration* menu
2. Click *Enable read-only mode*

 Anything that would require a change to a database will fail during read-only mode.

The button becomes *Disable read-only mode* when enabled. A banner appears above the main toolbar, notifying you that read-only mode is activated.

Disabling read-only mode, by clicking the *Disable read-only mode* button returns the server to its original, writable state.

Base URL Creation

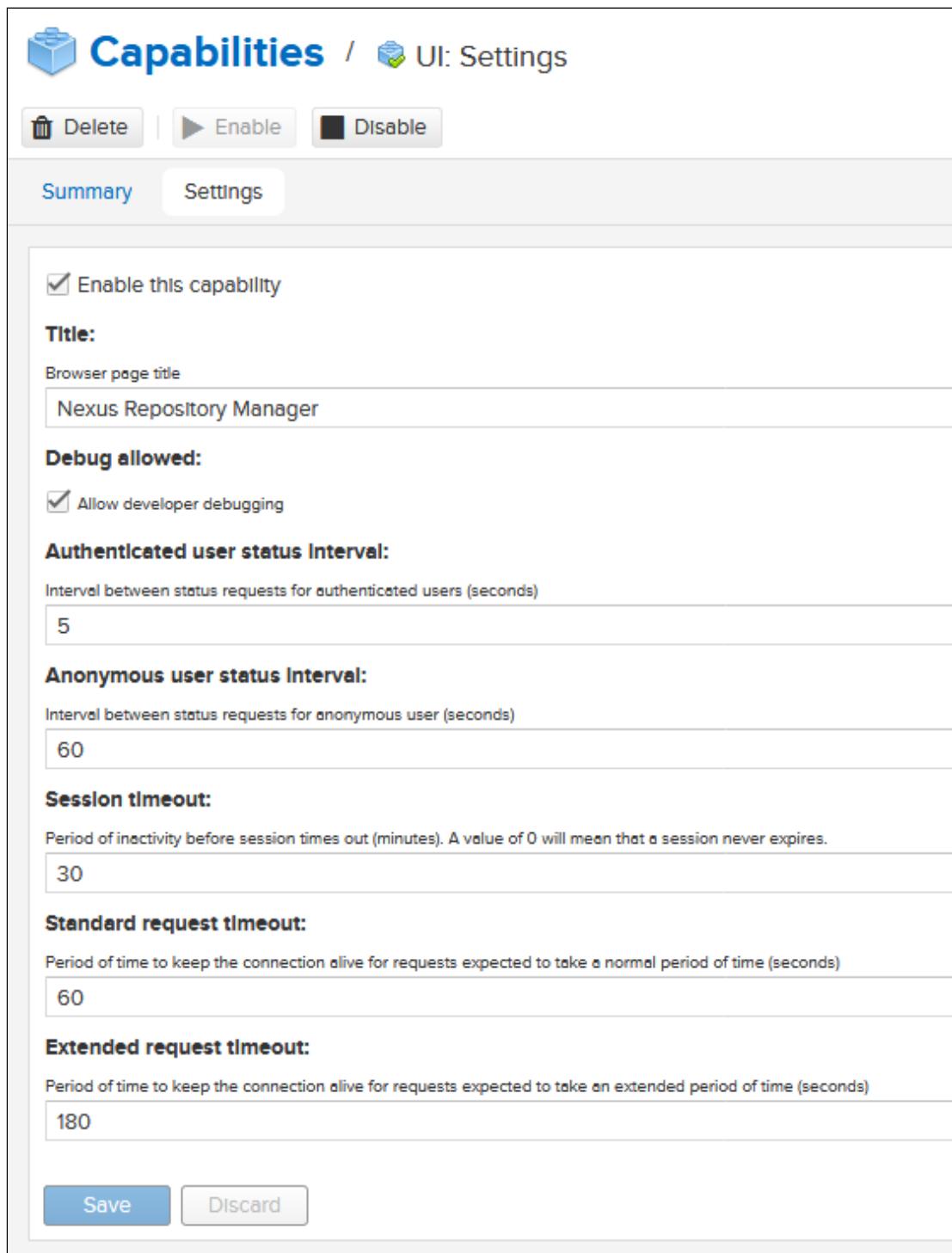
The *Base URL* is the address end users apply when navigating to the user interface. The repository manager only uses this value to construct absolute URLs to your user interface inside of email notifications. The most common reason why the address would be different is if you have a reverse proxy that terminates HTTP requests at an address different from where the repository manager is running.

To set the Base URL:

- Go to the *System* section of the *Administration* menu and select *Capabilities*.
- Search for an existing *Base URL* capability and select it if it exists or click the *Create capability* button and choose *Base URL* from the *Select Capability Type* list.
- Enter a new URL value.
- Press the *Create capability* to add the *Base URL*.

UI Settings

The *UI Settings* allow you to configure a few settings around UI usage, via a capability as displayed in Figure: “*UI Setting Capability Configuration*”.



The screenshot shows the 'Capabilities' section of the Nexus Repository Manager 3 interface, specifically the 'UI: Settings' capability. The page has a header with a delete, enable, and disable button. Below the header are two tabs: 'Summary' (selected) and 'Settings'. The main content area contains several configuration options:

- Enable this capability:** A checked checkbox.
- Title:** A field containing 'Browser page title' and 'Nexus Repository Manager'.
- Debug allowed:** A checked checkbox.
- Authenticated user status Interval:** A field containing '5'.
- Anonymous user status Interval:** A field containing '60'.
- Session timeout:** A field containing '30'.
- Standard request timeout:** A field containing '60'.
- Extended request timeout:** A field containing '180'.

At the bottom are 'Save' and 'Discard' buttons.

Figure: UI Settings Capability Configuration

Title

The text that will be displayed in the browser as the page title.

Debug allowed

Whether the ?debug flag will be allowed, to put the User Interface in debug mode.

Authenticated user status interval

The number of seconds to wait between sending status requests to the server.

Anonymous user status interval

The number of seconds to wait between sending status requests to the server for non-authenticated users.

Session timeout

The number of minutes to wait before expiring the UI session of an inactive user.

Standard request timeout

The number of seconds to allow a request to process before terminating the connection.

Extended request timeout NEW IN RELEASE 3.15

The number of seconds to allow long running requests to process before terminating the connection.

This applies to certain requests that may take considerably longer than others, including the upload of assets as well as generating a support zip.

HTTP and HTTPS Request and Proxy Settings

The repository manager uses HTTP requests to fetch content from remote servers. In some cases a customization of these requests is required. Many organizations use proxy servers for any outbound HTTP network traffic. The connection to these proxy servers from the repository manager needs to be configured to allow it to reach remote repositories. All this can be configured in the *HTTP* configuration available via the *System* section of the *Administration* menu and displayed in *Figure: “Configuring HTTP Request Settings”*.

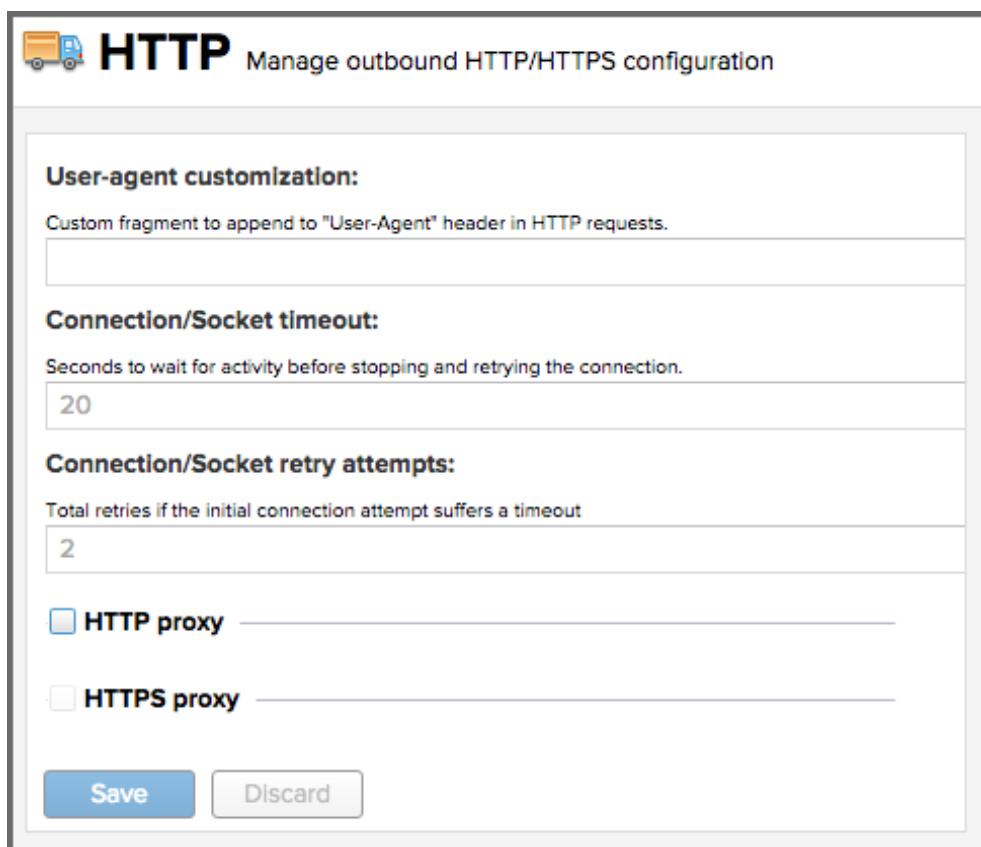


Figure: Configuring HTTP Request Settings

User-agent customization

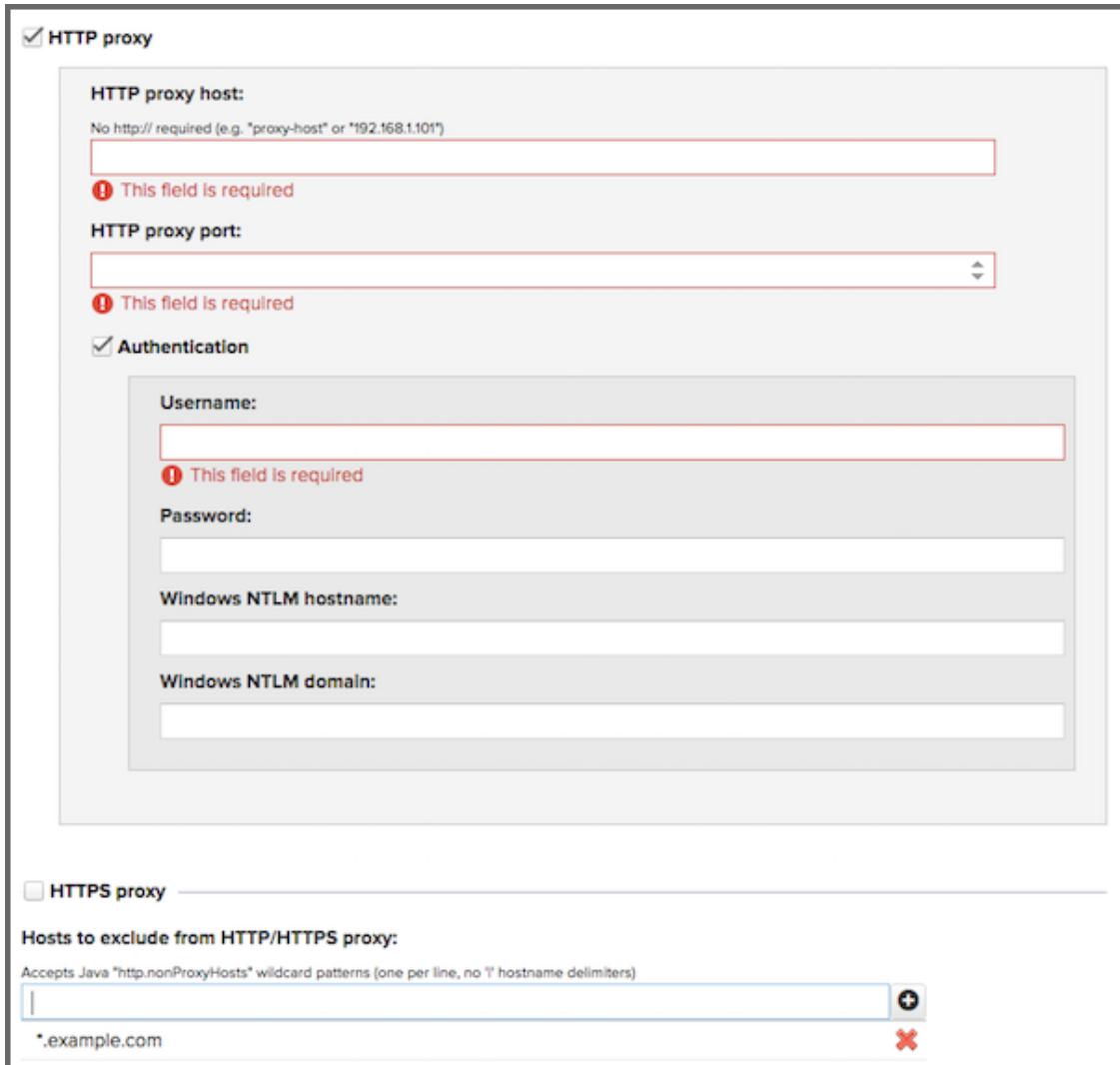
The HTTP configuration in *User-agent customization* allows you to append a string to the User-Agent HTTP header field. This can be a required customization by your proxy servers.

Connection/Socket timeout and attempts

The amount of time in seconds the repository manager waits for a request to succeed when interacting with an external, remote repository as well as the number of retry attempts to make when requests fail can be configured with these settings.

If your repository manager instance needs to reach public repositories like the Central Repository via a proxy server, you can configure the connection to a proxy server. Typically such an internal proxy server proxies HTTP as well as HTTPS connections to external repositories. In this case you configure a *HTTP proxy*. Select the checkbox beside *HTTP Proxy* and configure the parameters in the sections displayed in *Figure: "Configuring HTTP Proxy Settings"*. If your organization uses a separate, additional proxy server for HTTPS connections, you have to configure it in the *HTTPS Proxy* section.

- ✓ This is a critical initial step for many Enterprise deployments of Nexus Repository Manager Pro and Nexus Repository Manager OSS, since these environments are typically secured via an HTTP/HTTPS proxy server for all outgoing internet traffic.



The screenshot shows the 'HTTP proxy' configuration page. At the top, there is a checked checkbox labeled 'HTTP proxy'. Below it, there are two required fields: 'HTTP proxy host' and 'HTTP proxy port', both of which have red error messages indicating they are required. Underneath these fields is a checked checkbox labeled 'Authentication'. This section contains four fields: 'Username' (with a red error message), 'Password' (empty), 'Windows NTLM hostname' (empty), and 'Windows NTLM domain' (empty). At the bottom of the main configuration area, there is an unchecked checkbox labeled 'HTTPS proxy'. Below this, there is a section titled 'Hosts to exclude from HTTP/HTTPS proxy:' with a text input field containing '.example.com' and a delete button.

Figure: Configuring HTTP Proxy Settings

You can specify the *HTTP proxy host* and the *HTTP proxy port* of the HTTP or HTTPS proxy server and, optionally, the *Authentication* details for *Username* and *Password*. If a Windows NT LAN Manager is used to authenticate with the proxy server you can configure the needed connection details in *Windows NTLM hostname* and *Windows NTLM domain*.

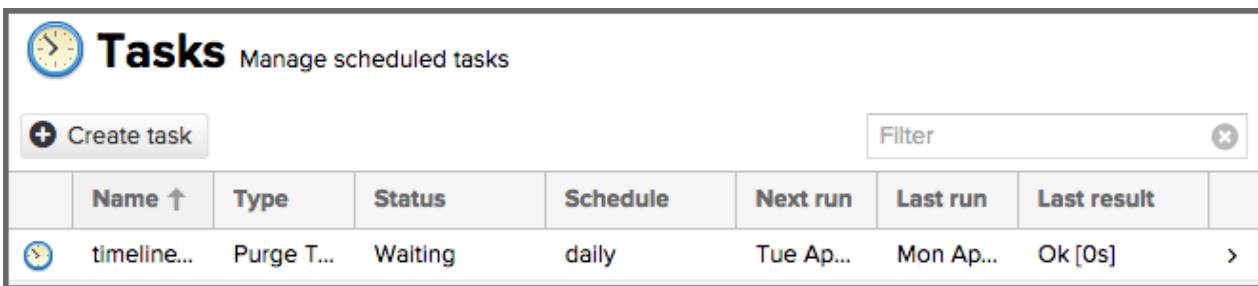
In addition, you can configure a number of hosts that the repository manager reaches directly, ignoring the proxy settings. Requests to them should not go through the configured HTTP/HTTPS proxy. These hosts can

be configured in the *Hosts to exclude from HTTP/HTTPS proxy* setting. You can add a hostname in the input box and add it with the + button. The * character can be used for wildcard matching for numerous host names allowing a setting such as *.example.com. Entries can be removed with the x button.

*Figure: “Configuring HTTP Proxy Settings” shows the *HTTP Proxy* administration interface. The *HTTPS* configuration interface looks the same and is found below the *HTTP* configuration.*

Configuring and Executing Tasks

The repository manager allows you to schedule the execution of maintenance tasks. The tasks can carry out regular maintenance steps that will be applied to all repositories or to specific repositories on a configurable schedule or simply perform other system maintenance. Use the *Tasks* menu item in the *System* section of the *Administration* menu to access the feature view, shown in *Figure: “Managing Tasks”*, that allows you to manage your Tasks.



Tasks <small>Manage scheduled tasks</small>									
		Name ↑	Type	Status	Schedule	Next run	Last run	Last result	
	timeline...	Purge T...	Waiting	daily	Tue Ap...	Mon Ap...	Ok [0s]	>	

Figure: Managing Tasks

The list interface allows you to add new tasks with the *Create task* button as well as inspect and work with the configured tasks. The list shows the following columns:

Name

A user-defined name for the task to identify it in the user interface and log files.

Type

The type of action the scheduled task executes. The list of available task types is documented in more detail below.

Status

Tasks can either be *Waiting* for their next run, currently *Running* or *Disabled*.

Schedule

The *Schedule* column shows the *Task frequency* e.g., *Daily*, *Monthly*, *Manual*, and others.

Next run

This column displays date and time of the next execution of the task based on the configured schedule.

Last run and Last result

These columns display the date and time as well as the result and duration of the last execution of the specific task.

When creating or updating a scheduled task, you can configure the following additional properties:

Task enabled

Enable or disable a specific task with the checkbox.

Notification Email

Configure a notification email for the task execution. If a scheduled task meets the notification condition configured below, a notification email will be sent to the configured email recipient. In the case of a failure, the notification will contain the task identifier and name as well as the stack trace of the failure. Otherwise, a summary of the task execution, including the start time and duration, will be sent.

Notification Condition NEW IN RELEASE 3.22

Notifications can be sent on *Failure* or on *Success or Failure*. If *Failure* is selected, a notification email will be sent only if the task ends abnormally.

Task frequency

Selecting the task frequency allows you to configure the schedule for the task executions. Available choices are *Manual*, *Once*, *Hourly*, *Daily*, *Weekly*, *Monthly*, and *Advanced* (provide a CRON expression). Apart from *Manual*, all choices trigger display of a custom user interface for scheduling the specific recurrence. Weekly scheduling requires at least one day of the week to be selected. The advanced setting allows you to provide a CRON expression to configure more complex schedules.

The syntax used for Advanced (provide a CRON expression) follows the UNIX-style CRON syntax. CRON expressions are comprised of 6 required fields and one optional field separated by white space as described in *Table 4.1, “Fields of a CRON Expression from Left to Right”* and the following paragraphs. A simple expression example is `0 0 9 * * ?`. This configuration triggers a task execution every day at 9:00 in the morning. Further examples are available in *Table 4.2, “CRON Expression Examples”*.

Field Name	Allowed Values	Allowed Special Characters
Seconds	0-59	, - * /
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day-of-Week	1-7 or SUN-SAT	, - * ? / L #

Field Name	Allowed Values	Allowed Special Characters
Year (Optional)	empty, 1970-2099	, - * /

Table 4.1. Fields of a CRON Expression from Left to Right

*

This character is used to specify any value. For example, * in the minute field means every minute.

?

This character is allowed for the day-of-month and day-of-week fields. It is used to specify no specific value. This is useful when you need to specify something in one of the two fields, but not the other.

-

This character is used to specify ranges. For example 10-12 in the hour field means "the hours 10,11 and 12".

,

This character is used to specify additional values. For example MON,WED,FRI in the day-of-week field means "the days Monday, Wednesday, and Friday".

/

This character is used to specify a start value and increments. For example "0/15" in the seconds field means "the seconds 0, 15, 30, and 45". And "5/15" in the seconds field means "the seconds 5, 20, 35, and 50". Specifying * before the / is equivalent to specifying 0 as the value to start with. Essentially, for each field in the expression, there is a set of numbers that can be turned on or off. For seconds and minutes, the numbers range from 0 to 59. For hours 0 to 23, for days of the month 0 to 31, and for months 1 to 12. The / character simply helps you turn on every "nth" value in the given set. Thus "7/6" in the month field only turns on month "7", it does not mean every 6th month, please note that subtlety.

L

This character is allowed for the day-of-month and day-of-week fields. This character is short-hand for "last", but it has different meaning in each of the two fields. For example, the value L in the day-of-month field means "the last day of the month" - day 31 for January, day 28 for February on non-leap years. If used in the day-of-week field by itself, it simply means 7 or SAT. But if used in the day-of-week field after another value, it means "the last xxx day of the month" - for example 6L means "the last Friday of the month". When using the L option, it is important not to specify lists, or ranges of values, as you will get confusing results.

W

This character is allowed for the day-of-month field. This character is used to specify the weekday (Monday-Friday) nearest the given day. As an example, if you were to specify "15W" as the value for the day-of-month field, the meaning is: "the nearest weekday to the 15th of the month". So if the 15th is a Saturday, the trigger will fire on Friday the 14th. If the 15th is a Sunday, the trigger will fire on Monday

the 16th. If the 15th is a Tuesday, then it will fire on Tuesday the 15th. However if you specify 1W as the value for day-of-month, and the 1st is a Saturday, the trigger will fire on Monday the 3rd, as it will not jump over the boundary of a months days. The W character can only be specified when the day-of-month is a single day, not a range or list of days.

- ✓ The L and w characters can also be combined for the day-of-month expression to yield LW, which translates to "last weekday of the month".

#

This character is allowed for the day-of-week field. This character is used to specify "the nth" XXX day of the month. For example, the value of "6#3" in the day-of-week field means the third Friday of the month (day 6 = Friday and #3 is the 3rd one in the month). Other examples: 2#1 is the first Monday of the month and 4#5 is the fifth Wednesday of the month. Note that if you specify #5 and there is not 5 of the given day-of-week in the month, then no firing will occur that month.

- ⚠ The legal characters and the names of months and days of the week are not case sensitive.

Expression	Description
0 0 12 * * ?	Fire at 12pm (noon) every day
0 15 10 ? * *	Fire at 10:15am every day
0 15 10 * * ?	Fire at 10:15am every day
0 15 10 * * ? *	Fire at 10:15am every day
0 15 10 * * ? 2015	Fire at 10:15am every day during the year 2015
0 * 14 * * ?	Fire every minute starting at 2pm and ending at 2:59pm, every day
0 0/5 14 * * ?	Fire every 5 minutes starting at 2pm and ending at 2:55pm, every day

Expression	Description
0 0/5 14,18 * * ?	Fire every 5 minutes starting at 2pm and ending at 2:55pm, AND fire every 5 minutes starting at 6pm and ending at 6:55pm, every day
0 0-5 14 * * ?	Fire every minute starting at 2pm and ending at 2:05pm, every day
0 10,44 14 ? 3 WED	Fire at 2:10pm and at 2:44pm every Wednesday in the month of March.
0 15 10 ? * MON-FRI	Fire at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday
0 15 10 15 * ?	Fire at 10:15am on the 15th day of every month
0 15 10 L * ?	Fire at 10:15am on the last day of every month
0 15 10 ? * 6L	Fire at 10:15am on the last Friday of every month
0 15 10 ? * 6L 2002-2005	Fire at 10:15am on every last Friday of every month during the years 2002, 2003, 2004 and 2005
0 15 10 ? * 6#3	Fire at 10:15am on the third Friday of every month

Table 4.2. CRON Expression Examples

The *Start date* and *Start time* allow you to configure a specific date and time from when the schedule should be activated. The *Time to run this task* settings is used to configure the actual time of the task execution.

Task-type specific configuration is displayed below the notification email input field and differs for each scheduled task.

Types of Tasks and When to Use Them

The following tasks are available to perform specified maintenance:

Task Name	Task Id	Description	Typically Scheduled

Admin - Cleanup Tags (Pro Only)	tags.cleanup	<p>This task will allow you to cleanup tags once they reach a certain age, haven't been modified in a certain period, or using a regular expression on the name.</p> <p>You can also choose to delete the components along with those tags, as well as restrict this component deletion to a particular format or repository.</p> <p>See additional details on the tagging page.(see page 363)</p>	Yes
Admin - Compact blob store	blobstore.compact	<p>Content deleted from a blob store(see page 235) is not physically deleted from the storage device. Instead it is only internally marked for deletion. This task performs the actual deletion of the relevant files, and therefore frees up the storage space.</p> <p>i This only affects blob store of type <i>File</i>. S3 type blob stores are cleaned up using AWS Lifecycle.</p>	Yes
Admin - Delete orphaned API keys	security.purge-api-keys	<p>This task deletes old, unused API keys. These keys are generated, for example, when using the User Token feature and become orphaned when the associated user account is deleted.</p>	Yes
Admin - Export databases for backup	db.backup	<p>This task performs a full backup⁶⁸⁶ of the underlying databases, including access logs, repository manager configuration, and security configuration. You must choose a location for the backup data for this task. When you run the backup, the task adds a timestamp to the backup files that are created in the backup data location. It is important to note this task does not backup actual repository content.</p>	Yes
Admin - Execute script	script	<p>Scripts can be provided in the Source field and have to be written using the Groovy programming language. These scripts can use the APIs of the repository manager to perform maintenance and other modification tasks. Please consult related documentation for writing scripts(see page 502) and additional information/references to the Javadoc and source.</p>	Optional

⁶⁸⁶ <https://help.sonatype.com/display/NXRM3M/Backup+and+Restore>

Admin - Log database table record counts (Pro Only)	cluster.periodicLogging	<p>This task logs clustered database record counts for each node. The task is automatically added when clustering is enabled and can be disabled by including an additional property in the <code>nexus.properties</code> file:</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <pre>nexus.log.cluster.enabled=false</pre> </div>	Yes
Admin - Remove a member from a blob store group	blobstore.group.memberRemoval	<p>This task removes a blob store from a group. Each blob within the removed store is written back to the group and then deleted from the removed blob store.</p>	No
Docker - Delete incomplete uploads	repository.docker.upload-purge	<p>This task cleans up orphaned files that may exist in temporary storage as result of a restart or incomplete/interrupted uploads.</p> <p>The <i>Age in hours</i> parameter allows you to configure the minimum age of incomplete uploads to be deleted.</p>	Yes
Docker - Delete unused manifests and images	repository.docker.gc	<p>This task will handle deletion of content that is no longer referenced, images that are no longer referenced by a tagged manifest and V1 layers that are no longer referenced by a tagged layer.</p>	Yes

Maven - Delete SNAPSHOT**	repository.maven.remove-snapshots	<p>This task can be scheduled to delete SNAPSHOT components from a Maven repository. Typically this is useful to preserve storage space, as old SNAPSHOT versions are not accessed after deployment of a new snapshot and no longer add value. When you create a scheduled task to delete snapshots, you can specify the repository to affect, as well as set the following parameters:</p> <p><i>Minimum snapshot count:</i> This configuration option allows you to specify a minimum number of snapshots to preserve per component SNAPSHOT version.</p> <p><i>Snapshot retention (days):</i> This configuration option allows you to specify the number of days to retain component SNAPSHOT versions.</p> <p><i>Remove if released:</i> If checked, all SNAPSHOT versions that match any released component found with the same groupId and artifactId coordinates will be removed.</p> <p><i>Grace period after release (days):</i> This parameter allows you to specify a number of days before released snapshots are purged. If a release associated to a snapshot has an updated timestamp and falls within the set grace period, it will not be deleted. This setting will give the respective project that references the snapshot dependency time to upgrade to the release component or the next snapshot version.</p> <p>The deletion of Maven snapshots when <i>Remove if released</i> is checked takes precedence over the number you select in the <i>Minimum snapshot count</i> field. NOTE: It is possible to configure the task in such a way that the results may be unexpected. If configured to keep 0 minimum snapshots older than 0 days, all snapshots everywhere will be deleted, despite whether or not a grace period is configured for releases.</p>	Yes
---------------------------	-----------------------------------	---	-----

Maven - Delete unused SNAPSHOT**	repository.maven.purge-unused-snapshots	<p>This task can be used to delete unused SNAPSHOT files from Maven repositories. Any SNAPSHOT that has not been requested in the configured number of days will be purged.</p> <p>i In the event the component has never been requested this task will use the uploaded date instead.</p>	Yes
Maven - Publish Maven Indexer files	repository.maven.publish-dotindex	The task publishes the indexer files for all or a specific Maven repository.	Optional
Maven - Unpublish Maven Indexer files	repository.maven.unpublish-dotindex	This task is the counterpart to the task <i>Maven - Publish Maven Indexer files</i> and can remove the indexer files.	Optional
Repair - Rebuild Maven repository metadata (maven-metadata.xml)	repository.maven.rebuild-metadata	This task rebuilds the maven-metadata.xml files with the correct information and will also (optionally) validate and fix any incorrect checksums (.md5/.sha1) for all files in the specified maven2 hosted repository. The <i>Group Id</i> , <i>Artifact Id</i> and <i>Base Version</i> parameters allow you to narrow down the section of the repository that will be repaired. Typically this task is only run manually to repair a corrupted repository.	No
Repair - Rebuild repository browse	create.browse.nodes	This task rebuilds the tree browsing data based upon current information in the database.	No
Repair - Rebuild repository search	repository.maven.rebuild-metadata	<p>With support for hosted and proxy repositories, this task rebuilds the search index. It inspects actual components and assets found in the selected repository and thus reflects the true content for supporting search and browse actions.</p> <p>i Cancelling this task before completion could result in a partially rebuilt search index showing incomplete component search results until it is run again and allowed to complete.</p>	No

Repair - Reconcile component database from blob store	blobstore.rebuild ComponentDB	<p>This task allows you to recover lost asset/component metadata from a chosen blob store. The task is useful in cases where you want to recover lost information from corrupt databases. When executed, the task searches the selected blob store for blobs missing their associated metadata. The asset/component metadata is restored based on the information contained in the blob store.</p> <p>Currently, this task only recovers metadata for Apt, Docker, Go, Helm, Maven, npm, P2, PyPI, R, Raw, RubyGems and Yum repositories.</p> <div style="border: 1px solid red; padding: 10px;"> <p>! npm and Yum</p> <p>npm and Yum search and tree view browsing will not work after running this task unless you run the respective metadata repair tasks (<i>Repair - Reconcile npm /-/v1/searchmetadata</i> and <i>Repair - Rebuild Yum repository metadata (repodata)</i>) first. Builds will work however.</p> <p>Running these tasks before restore is complete will likely cause incomplete search and browse results.</p> <p>NOTE: To avoid resultant possible errors in search and browse, we recommend running <i>Repair - Rebuild repository browse</i> and <i>Repair - Rebuild repository search</i> tasks when your component database repair is complete.</p> </div>	No
Repair - Reconcile date metadata from blob store	rebuild.asset.uploadMetadata	This task reads the blobstore and updates the assets in the database with date information.	No

Repair - Reconcile npm /-/ v1/ search metadata	repository.npm.reindex	This task extracts search metadata from all npm packages in all npm hosted repositories to enable support for the new /v1/ search endpoint that replaced the /all/ endpoint.	No
Repair - Rebuild Yum repository metadata (repodata)	repository.yum.rebuild.metadata	This task rebuilds the metadata for a chosen Yum hosted repository. This task runs automatically 60 seconds (configurable) after an RPM is uploaded, deleted or redeployed.	No
Repository - Delete unused components**	repository.purge-unused	This task can be used to remove components and assets in proxy repositories. Any component that has not been requested in the configured number of days will be purged.	Yes

i NOTE: Tasks deleting components only removes metadata for the components and assets they affect, they do not reclaim disk space used by the binary assets. This is achieved by using the *Admin - Compact blob store* task afterward.

i Tasks with the prefix "Repair -" are only intended to be run if you are having specific trouble with your system. These tasks should only be run manually, not scheduled, and should usually only be run at the advice of a Sonatype staff member.

Beyond these tasks, any plugin can provide additional scheduled tasks, which will appear once you have installed the plugin.

Setting up task execution adapted to your usage of the repository manager is an important first step when setting up a Nexus Repository Manager instance. It is also important to update your tasks when changing your usage patterns. For example, adjustments to your tasks may be required if you start to regularly deploy snapshots by introducing continuous integration server builds with deployment.

Task Logging

The output of every task run will go to a separate log file. By default these task logs are stored in `$data-dir/log/tasks`. The file name of each task log is the type followed by the full date and time the task started. For example: `repository-maven.purge-unused-snapshots-20170618153235.log`.

Generally the output of the task will go to both the `nexus.log` and the specific task log, however some tasks will only go to the `nexus.log`.

For long running tasks, progress will be logged back to the `nexus.log` every 10 minutes as the work continues. Most commonly this will appear as a contextual update relevant to that particular task such as the number of items that have been processed, updated, deleted, etc.

Task log files are removed after 30 days.

License Management

 **Available in Nexus Repository Pro only**

A paid license is necessary to upgrade Nexus Repository Manager OSS to Nexus Repository Manager Pro, and to keep Nexus Repository Manager Pro paid features operational. You must be logged in as a user with sufficient privileges to manage licenses.

Uploading a License

The paid license is a special `.lic` file that you upload to the *Licensing* feature view, found in the *Administration* → *System* → *Licensing* menu. To install the license:

- Upload the file from the *Select license...* button.
- Select the correct `.lic` file in the file selection dialog, and press *Open*.
- Click the *Install license* button.
- Enter your password when the re-authentication dialog appears.
- Click *I agree* to the terms stated in the End User License Agreement.
- Click *Authenticate* to complete the upload.

After the file is successfully uploaded, you will receive a notification to restart the repository manager. After restart the *License type*, shown in the Licensing panel, will display the features associated with your license.

Monitoring Recent Connections for Product License Compliance

If you wish to approximate the number of unique users accessing repository manager, it is recommended to use a log parser on the `request.log` files. The IP address and authenticated user ID associated with each request will be in columns 1 and 3 of each log line respectively.

Using LDAP queries against your LDAP server is another method that can help verify the number of unique users configured to access or are potentially accessing an NXRM instance.

See our [knowledge base article](#)⁶⁸⁷ for more information.

Recent Connections UI

REMOVED IN VERSION 3.17.0

Admin users can generate a record of users who had sessions within the last 7 days in the repository manager. In the *Administration* menu, go to the *Licensing* → *Recent Connections* to access the table.

The table displays the IP address (*IP*), last accessed date (*Date*), user name (*User*), and client (*User agent*). To generate the report click *Download* to produce a CSV file of listed users.

When a Nexus Repository Manager Pro license expires all functionality will be disabled in the user interface, except for the ability to install a new license file. To avoid interruption of service be sure to upload a renewed license before the existing license expires. Nexus Repository Manager Pro will provide a warning when the license is within 30 days of expiry.

Storage Guide

Introduction

A blob store is the storage for components and assets. Each blob store can be used by one or multiple repositories and repository groups. By default, a file blob store named *default* is automatically created in the directory `$data-dir/blobs/default` during installation. New blob stores may be configured by selecting *Blob Stores* from the *Repository* sub menu of the *Administration* menu. Once a blob store is created, many of its attributes are immutable. This makes it important to carefully plan your blob store configuration.

- ⓘ Learn the basics of storage management⁶⁸⁸ in our technical guide.

Terminology

- **Blob** - An object containing data within a blob store, e.g. component binaries and metadata files
- **Blob Store** - An internal storage mechanism for the binary parts of components and their assets
- **Concrete Blob Store** - Any blob store that is not a group
- **Fill Policy** - For a *Group Blob Store*, the fill policy determines which blob store a blob is written

⁶⁸⁷ <https://support.sonatype.com/hc/en-us/articles/213464168-Sonatype-Nexus-License-FAQ>

⁶⁸⁸ https://guides.sonatype.com/repo3/technical-guides/storage-primer-nxrm3/?utm_source=nxrm3-help-docs&utm_medium=HSC&utm_campaign=guides-promo

- **Group Blob Store** - A blob store that delegates operations to one of the other blob stores on its list
- **Hard Delete** - When a soft deleted blob is permanently removed from the blob store, usually via the *Admin - Compact blob store* task
- **Promote to Group** - A process by which a concrete blob store becomes a group blob store that can access the blobs of the original concrete blob store
- **Soft Delete** - Blobs deleted in a repository are only marked for deletion. They still exist within the blob store. That is known as a soft delete.
- **Soft Quota** - A check of a blob store that compares a metric against a limit. If the check fails, then the blob store is considered unhealthy, but writes still proceed and a warning is logged.
 - **Space Limit** - A byte limit compared against specific blob store metrics by a soft quota.
 - **Space Used Quota** - This type of quota is violated when the total size of the blob store exceeds the space limit.
 - **Space Remaining Quota** - This type of quota is violated when the available space of a blob store falls below the space limit.

Fields

- **Available Space** - Remaining storage capacity of a blob store
- **Blob Count** - The number of blobs currently stored in a blob store
- **Blob Store Type** - The type of blob store implementation
 - **File** - Store blobs in file system-based storage
 - **Group** - Combines multiple blob stores into one
 - **S3** - Store blobs in AWS S3 cloud storage

Planning

ⓘ If you have a High Availability cluster, then you should refer to this guide as well, [Configuring Blob Stores](#)(see page 534).

You will need to choose which types to use, how many blob stores to create, and how you allocate repositories to these blob stores. These decisions should be based on:

- the size of your repositories
- the rate at which you expect them to grow over time
- the storage space available
- the options you have available for adding storage space

Blob Store Types

Nexus Repository Manager supports several types of blob stores. File blob store is the default and is recommended for most installations.

File Blob Store

A file blob store lets Nexus Repository Manager store blobs as files in a directory. The location of the blob files is determined by the *Path* parameter supplied when creating the blob store. The *Path* can be on either a local disk or a NFSv4 compatible mount. Our guide, [Configuring Blob Stores](#)(see page 534), has more in depth information about configuring storage services for use with blob stores.

 Nexus Repository Manager does not support the following file systems types:

- NFS versions 3 and older
- GlusterFS
- FUSE based user space file systems

S3 Blob Store

An S3 blob store saves blobs as objects within a bucket on AWS S3.

 Requirements

- The S3 blob store is only recommended for Nexus Repository Manager installations hosted in AWS.
- The S3 blob store should be in the same AWS region as the Nexus Repo installation. Using different regions will result in unacceptably slow performance.
- Nexus Repository Manager only supports AWS S3. Other implementations of the S3 protocol may or may not work, and since they are not tested, they are not supported.

Nexus Repository Manager will automatically create an S3 bucket when a blob store is created, if it does not already exist. However, you may also create the bucket yourself. Note that:

- Nexus Repository Manager will automatically apply a lifecycle rule to expire deleted content.
- The bucket can use server-side encryption with KMS key management transparently. Other methods of server side encryption are not supported.
- If running on EC2 instances, Nexus Repository Manager can use the IAM role assigned to those instances when accessing S3 buckets.

Carefully consider whether S3 is the right storage solution for you. The performance is highly dependent on the speed of the network between NXRM and the AWS endpoint you connect to. NXRM will send multiple outbound HTTP requests to AWS to store blobs into S3, and large blobs are split into chunks over multiple requests. If your NXRM instance is not in AWS or connecting to another region, an S3 blob store may be significantly slower than a file based blob store.

For optimum performance:

- Run NXRM on AWS on EC2 instances
- Ensure that the S3 connection is using the region where NXRM is being run

The chunk size when uploading to S3 can be adjusted by setting the property `nexus.s3.multipartupload.chunksize` in the `nexus.properties` file. The default is set to 5 MB (also the minimum allowed value) and may be tuned for optimal performance on your network.

Group Blob Store

NEW IN 3.15.0 PRO

A group blob store combines concrete blob stores so that they act as a single blob store for repositories. A concrete blob store can either be used directly by a repository or as a member of a single group, but not both. Groups allow you to add members dynamically, but removing a blob store requires a task to ensure that repositories still have access to their blobs. More information about the task can be found in the *Admin - Remove a member from a blob store group* section of [Types of Tasks and When to Use Them](#)(see page 273). The *Fill Policy* is the method that the blob store group uses to choose a member for writing blobs. The *Fill Policy* can be changed at any time.

Available choices for *Fill Policy*:

- **Round Robin** - Incoming writes alternate between all blob stores in the group. If writing to one blob store fails, then it attempts again with the next blob store in the group. This is useful when you have a number of blob stores available and you want each of them to receive a roughly equal number of writes. This does not balance based upon any other metric.
- **Write to First** - All incoming writes are given to the first blob store. If you need to direct all blobs to a specific blob store, e.g. you have a blob store for a new empty disk, then this fill policy will ensure that the new blob store gets all the writes.

Blob Store Layouts

Once a repository is allocated to a blob store, it is there permanently. Blob stores cannot be split but they can be moved from one storage device to another using a manual process. The instructions to do so are in this support article, [Relocating Blob Stores](#)⁶⁸⁹. For these reasons, your approach to using blob stores should be chosen carefully

⁶⁸⁹ <https://support.sonatype.com/hc/en-us/articles/235816228-Relocating-Blob-Stores>

Single Blob Store

The simplest approach is to create a single blob store per storage device and divide your repositories among them. This is suitable in the following cases:

- Your repositories are growing slowly enough that you will not exceed your available storage within a year.
- If you exceed available storage, you will be able to move blob stores to larger storage devices.

Multiple Blob Stores

You should separate repositories into two or more blob stores per storage device if both:

- You expect to exceed your currently available storage within a year.
- You cannot move blob stores to larger storage devices, so you must add capacity to Nexus Repository Manager by adding additional storage devices.

In extreme cases of unpredictable capacity, the best approach is to create a separate blob store for each repository. This configuration adds significant complexity so should be considered with caution.

Group Blob Stores

NEW IN 3.15.0 PRO

Groups blob stores allow you to combine multiple blob stores into one. Groups are a good choice if:

- You need to add more storage space via multiple devices.
- You need the ability to spread writes and reads across multiple blob stores.

You can start with a concrete blob store and later use the *Promote to Group* process to transform it into a group for more flexibility. Promotion creates a group containing the previous blob store to which you can add other blob stores.

Estimating Storage Requirements for Blob Stores

Blob stores contain two files for each binary component stored in Nexus Repository Manager:

- The binary component, stored as a .bytes file (whose size is the same as the component)
- A properties file that stores a small amount of metadata for disaster recovery purposes (<1k)

The total storage size of a blob store is therefore approximately the total size of all of your components, plus an allowance for the properties files and the block size of your storage device. On average, the allowance will be $1.5 * \# \text{ of components} * \text{block size}$.

Example: Moving a Blob Store

The following steps can be used to move a blob store to a new location. This can also be used to change a blob store's storage, for instance, a file based blob store can be moved to s3.

1. Create a new blob store with the storage path set to the new location.
2. Then promote the existing blob store to a group blob store.
3. Set the new group's *Fill policy* to *Write to First*.
4. Add the new blob store (created in step 1) into the group blob store underneath the original blob store.

Then schedule and run an *Admin - Remove a member from blob store group* task via *Administration* → *System* → *Tasks* to remove the original blob store from the group. The contents of the original blob store will be moved over to the new blob store, and then the original blob store will be removed.

Auditing

Auditing is done using a capability, as described in [Accessing and Configuring Capabilities](#)(see page 259) called *Audit*. For your convenience, this capability is created and enabled by default.

When enabled, a log file located in \$data-dir/log/audit/audit.log will be updated each time a user (or internal processes) modify the configuration of NXRM, as well as any asset and component additions and removals. Each line of this log contains an unformatted JSON message representing a single audit item. Find a list of available attributes in these JSON messages below:

Attribute	Description	Example
timestamp	The date and time this event occurred	2019-02-04 18:12:07,856-0500
nodeId	The nodeId of the NXRM instance (used to correlate audit logs across multiple NXRM instances)	5DF0F434-88A6F4B7-AEDCE785-CAD9628C-8AD86154
initiator	The initiator of the event, often a username/host combination, unless an internally generated event	admin/172.16.0.10
domain	Functional area of the system	security.user
type	Action performed in this domain	created
context	Identifying details of the event	mynewusername
attributes	Map of key:value pairs that contain more details about the event	{"id": "mynewusername", "name": "John Doe", "email": "jdoe@emailserver.com", "source": "default", "status": "active", "roles": "nx-admin"}

This log file will rotate daily, and a maximum of 90 days worth of files will be retained.

 **Where is the Audit user interface?**

NEW IN VERSION 3.16.0

The database which stored audit events and the interface at *Administration* → *System* → *Audit* has been replaced by an audit log text file. The new implementation is better performing and more easily integrated with external tooling.

Routing Rules

As an administrator of a Nexus Repository Manager instance, you may want to prevent it from making certain requests to upstream repositories. This could be used to prevent leaking the names of internal projects to external registries, or to prevent a name hijacking attack where a malicious user creates packages in a registry with names used by your internal projects.

Creating or Modifying a Routing Rule

The *Routing Rules* section of the interface can be found in the *Administration* section under *Repository*. A user will need to have the `nx-all` privilege to view or make changes to routing rules. To edit a routing rule select it from the list or to create a new one click the *Create Routing Rule* button in the toolbar.

 **Routing Rules** /  Create Routing Rule

Name:
stop-leaks

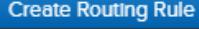
Description:
Prevent requests of internal names

Mode:
BLOCK ▾ requests when their path matches any of the following matchers

Matchers:
Regular expressions used to identify request paths that are allowed or blocked (depending on above mode)

/^com/example/.*
/^org/example/.*



Routing rules have two modes: *BLOCK* and *ALLOW*. When *BLOCK* is chosen if one of the matchers matches the request path then the request will be blocked, otherwise it will be allowed. If a rule uses *ALLOW* mode then the request path must be matched by at least one of the matchers or it will be blocked.

Each routing rule must define one or more matchers; these must be valid regular expressions as allowed by Java. Defining these may require some understanding of URLs used by formats; alternatively one could try to keep matchers simple and match your company name, for example `.*sonatype.*`. With regular expressions its possible to create matchers for a wide variety of scenarios, however, one should keep in mind that regular expressions can be expensive to evaluate and in extreme cases lead to a [Regular Expression Denial of Service](#)⁶⁹⁰.

Testing Routing Rules

Below the routing rule editor is a section to help test the rule. With this you could specify a possible request path (for example copying one from a search result or from the request log) which will verify whether the request would be allowed or blocked by the rule as written. As noted in the subtext, request paths will always begin with a leading slash which is supplied by the test tool automatically.

⁶⁹⁰ https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS

Test This Routing Rule

Request Path:

Requests always start with a leading slash

Test

Deleting a Routing Rule

Routing rules can be removed by using the *Delete Routing Rule* button from the toolbar while viewing a routing rule. In order for the delete to be successful the rule must not be assigned to any repositories.

Assigning a Rule to a Repository

While editing a proxy repository there is a section titled *Routing Rule* which allows the user to select a previously created routing rule to use for the repository. Entering text into the field will allow you to select the routing rule name from a filtered list of routing rules. Failing to choose one will clear out the field when it loses focus. Click the triangle on the right to display all options. For more information see [Repository Management](#)(see page 234).

Routing Rule

Choose a rule to restrict some requests from being served by this repository

None

None

N stop-leaks

Retry Limit Configuration

Introduction

The following system properties and JMX bean are provided to allow the user to control when and how often retries occur for storage transactions.

System Properties

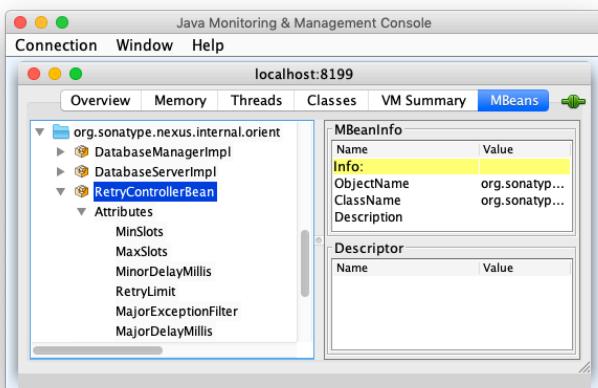
The following system properties can be used to control when and how often a retry occurs:

```
nexus.tx.retry.limit=8          # maximum times a transaction can retry
nexus.tx.retry.minSlots=2        # minimum number of backoff slots
nexus.tx.retry.maxSlots=256      # maximum number of backoff slots
nexus.tx.retry.minorDelay=10ms   # per-slot delay if it's a minor exception
nexus.tx.retry.majorDelay=100ms  # per-slot delay if it's a major exception

# comma-separated list of exceptions to treat as "major"
nexus.tx.retry.majorExceptionFilter=\
    java.io.IOException,org.sonatype.nexus.repository.storage.MissingBlobException
```

IOExceptions and MissingBlobExceptions are treated as major exceptions which result in longer backoff delays before retrying. This is because these exceptions are often related to slower (networked) storage which works best with a longer delay before retrying. When running in HA-C various distributed exceptions are also listed as major for the same reason.

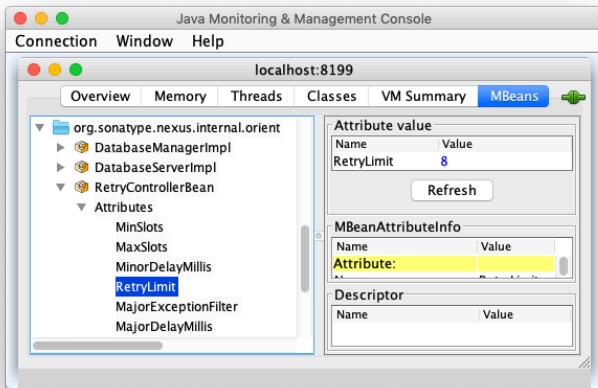
RetryControllerBean



This JMX bean can be used to tune the retry settings at runtime without rebooting.

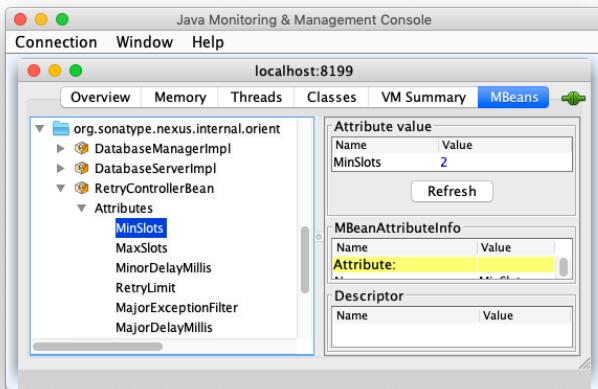
Attributes

RetryLimit



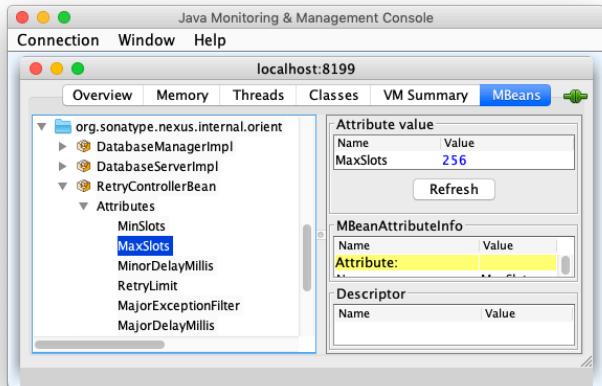
The maximum number of times a transaction can retry.

MinSlots



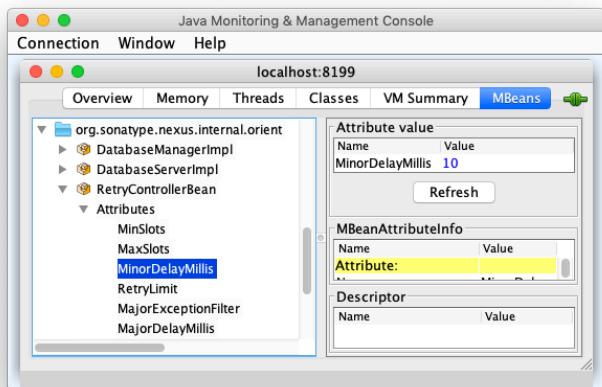
The minimum number of backoff slots.

MaxSlots



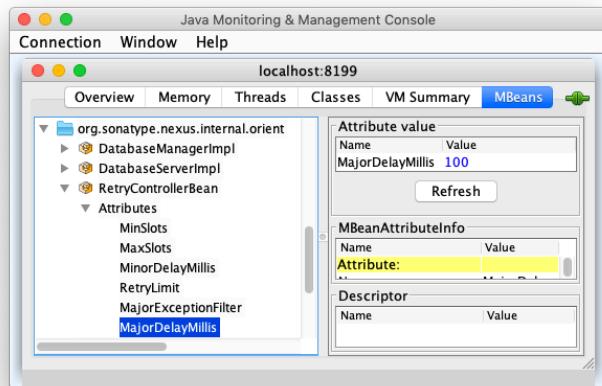
The maximum number of backoff slots.

MinorDelayMillis



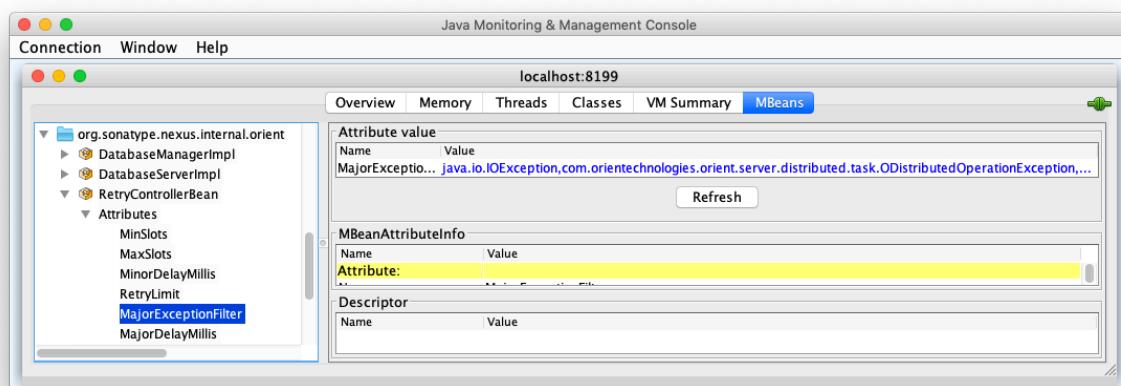
The per-slot delay if it's a minor exception.

MajorDelayMillis



The per-slot delay if it's a major exception.

MajorExceptionFilter



A comma-separated list of exceptions (fully qualified names) to treat as "major".

IOExceptions and MissingBlobExceptions are treated as major exceptions which result in longer backoff delays before retrying. This is because these exceptions are often related to slower (networked) storage which works best with a longer delay before retrying. When running in HA-C various distributed exceptions are also listed as major for the same reason.

References

- https://en.wikipedia.org/wiki/Exponential_backoff

Backup and Restore

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

Nexus Repository Manager lets you utilize a scheduled task to aid with backing up your repository manager. Along with your backup procedure you can configure your repository manager to save the OrientDB databases that store your component metadata and system configurations.

You can configure this task to export settings and metadata from the underlying OrientDB databases. When running the task it:

- stores the databases to a new location when configured.
- generates a snapshot of the databases for you to back up, along with the other parts of the repository manager.
- suspends access to the database until the backup is complete.
- cancels any currently running scheduled tasks.

It's recommended you develop a backup procedure and provide a backup location for your Nexus Repository Manager. Finally, you must synchronize the database exports with other parts of the repository manager that you wish to back up.

This section shows you how to configure and execute the tasks as well as to learn how to and recover the exported databases of your Nexus Repository Manager.

Topics in this section:

- [Prepare a Backup\(see page 294\)](#)
- [Configure and Run the Backup Task\(see page 295\)](#)
- [Restore Exported Databases\(see page 296\)](#)

Prepare a Backup

NXRM stores data in [blob stores\(see page 235\)](#). Metadata and configuration information are stored separately in databases. The blob stores and metadata databases must be backed up together or the blob store metadata may be non-descript when attempting to restore the exported data. Your backup strategy should involve backing up both your databases and blob stores together to a new location, in order to keep the data intact.

ⓘ Complete the following steps to perform a backup:

1. [Blob Store Backup](#)(see page 295)
2. [Database Backup](#)(see page 295)

Blob Store Backup

The filesystem or object store containing the blobs must be backed up outside of NXRM.

- For *File* blob stores, the directory storing the blobs should be backed up. For a typical configuration this will be \$data-dir/blobs.
- For S3 blob stores, bucket versioning can be used as an alternative to backups or the bucket can be mirrored to another S3 bucket.

Database Backup

The databases that you export have pointers to blob stores which contain components and assets potentially across multiple repositories. If you don't back them up together the component metadata can point to non-existent blob stores, or the blob store metadata may be non-descript when attempting to restore the exported data. So, your backup strategy should involve backing up both your databases and blob stores together to a new location, in order to keep the data intact.

Here's a common scenario for backing up custom configurations in tandem with the database export task:

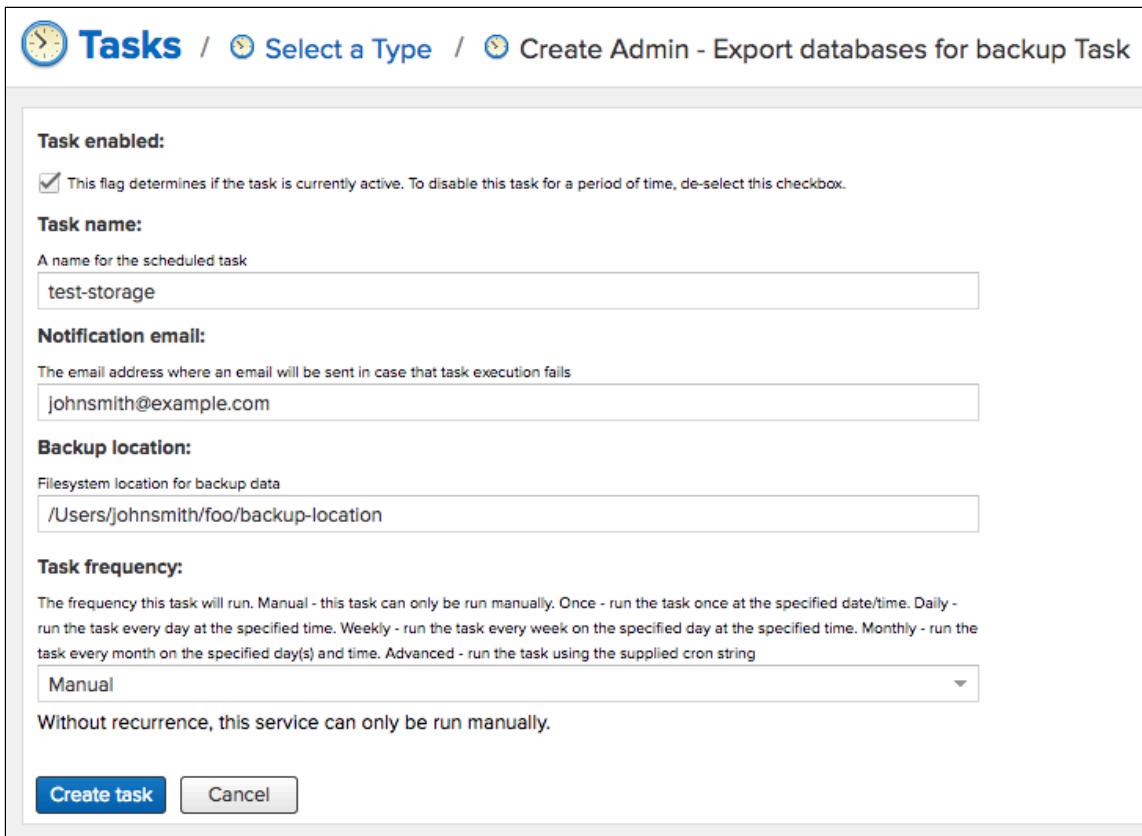
1. Configure the *Admin - Export databases for backup* Task to export databases.
2. Run the scheduled task to export the databases to the configured folder.
3. Back up custom configurations in your installation and data directories at the same time you run the export task.
4. Back up all blob stores.
5. Store all backed up configurations and exported data together.

⚠ Write access to databases is temporarily suspended until a backup is complete. It's advised to schedule backup tasks during off-hours.

Configure and Run the Backup Task

To configure and run a new task for database backup, review the steps in [Configuring and Executing Tasks](#)(see page 269). The form for this task includes an additional field called *Backup location*, which requires

you to enter the path to a directory where you want to store backup data, shown in *Figure: "Admin - Export databases for backup Task with Directory Location"*.



The screenshot shows a configuration form for a scheduled task. At the top, there are navigation links: 'Tasks' (with a clock icon), 'Select a Type' (with a gear icon), and 'Create Admin - Export databases for backup Task' (with a database icon). The main area contains several configuration sections:

- Task enabled:** A checkbox is checked, with a note below it: "This flag determines if the task is currently active. To disable this task for a period of time, de-select this checkbox."
- Task name:** A text input field containing "test-storage".
- Notification email:** A text input field containing "johnsmith@example.com".
- Backup location:** A text input field containing "/Users/johnsmith/foo/backup-location".
- Task frequency:** A dropdown menu set to "Manual", with a note below it: "Without recurrence, this service can only be run manually."

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

Figure: "Admin - Export databases backup Task with Directory Location"

When the task runs it exports backup data to the path specified in the *Backup location* field. The directory you input will contain .bak files of the following databases:

Component

all related data that make up components within the repository manager

Configuration

general administrative configurations such as scheduled tasks, email server configuration

Security

all user and access rights management content

All backup files are presented in the timestamp format based on the time the task was started.

Restore Exported Databases

You can restore the exported database files to the state when you ran the scheduled task. This requires you to:

- access the location specified in the *Backup location* field from the *Admin - Export databases for backup* task where the databases were exported
- remove all of the database directories in order to restore them to the previous state

The restoration should include all of the databases that were exported during the backup process. The databases should not be restored individually, and only files from a single backup (i.e. those with the same timestamp) should be used during restoration.

Start the database restoration with these steps:

1. Stop Nexus Repository Manager
2. Remove the following directories from \$data-dir/db
 - component
 - config
 - security
3. Go to the location where you stored the exported databases
4. Copy the corresponding .bak files to \$data-dir/restore-from-backup for restoration (**Note:** For version 3.10.0 or earlier use \$data-dir/backup as the restore location)
5. Restart Nexus Repository Manager

You can verify the restoration is complete by viewing the fully-restored databases previously removed from \$data-dir/nexus3/db.

 When the component database is restored, the corresponding blob stores containing components must also be restored.

Cleanup Policies

NEW IN RELEASE 3.14

Summary

The cleanup feature is a way of removing components from your repositories using policies customized to suit your needs. Cleanup policies can be shared between repositories. The [Cleanup Task](#)(see page 304), that performs the cleanup, executes at a frequency decided by you.

By default no policies are created or assigned to any repositories, however, it is highly recommended that an internal cleanup strategy is developed based on repository usage, available storage space and component size. It is further recommended that cleanup and related policies are determined before disk space issues arise and the situation becomes reactive.

Policy setup may vary by format or repository and very likely varies per installation.

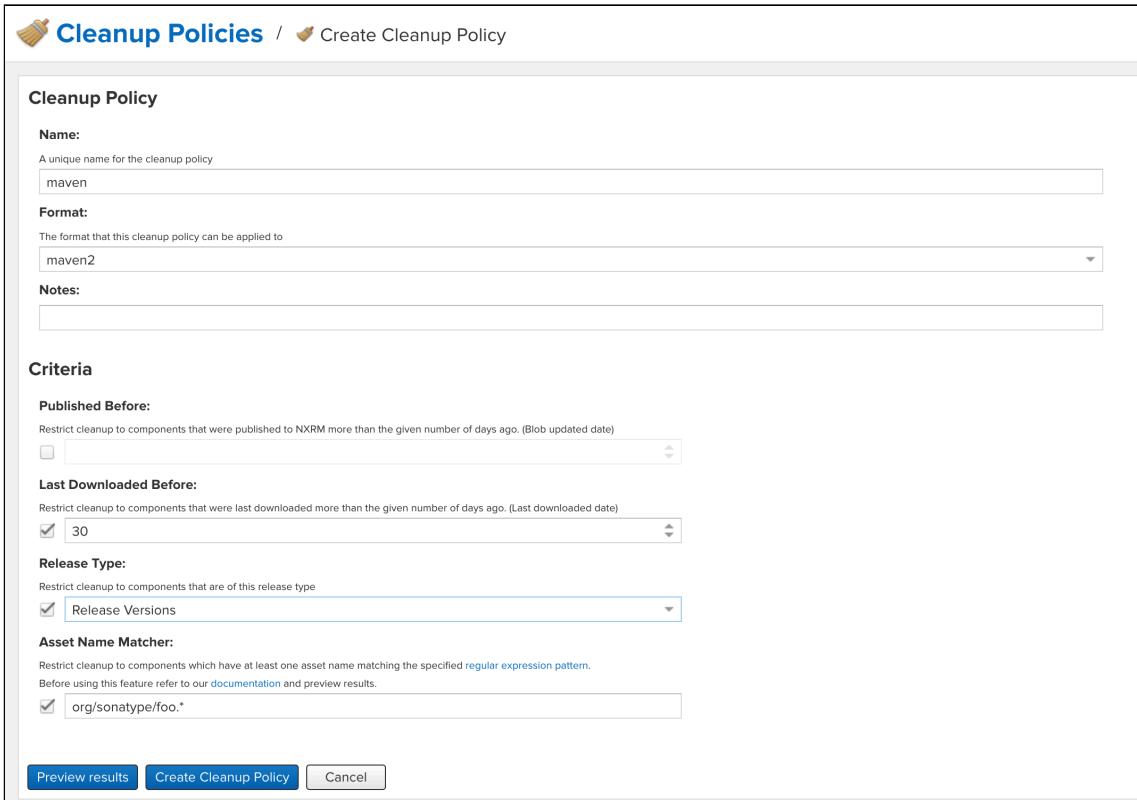
! Anything deleted by cleanup policies is soft deleted. Disk space is **not** reclaimed until an *Admin - Compact blob store* task is run. Frequency and/or automation of that task should be done to your level of caution. See [here](#)(see page 269) for more general information on tasks.

Despite the soft deletion, misconfiguration of a cleanup policy can result in the removal of most or even all of your components so caution (and use of preview) is recommended.

Cleanup Policies

A Cleanup Policy can be created for a given format, or optionally for all formats to be used generally, from the *Cleanup Policies* menu item in the *Repository* section of the *Administration* menu. Each policy's criteria are ANDed together removing only components that meet all of the specified conditions (e.g. items that arrived before X days AND were downloaded before Y days).

The creation screen is shown below. The *Name* and *Format* are required fields and not editable once created. Any criteria enabled is required but is also editable.



The screenshot shows the 'Create Cleanup Policy' dialog box. At the top, there are two tabs: 'Cleanup Policies' (selected) and 'Create Cleanup Policy'. The main area is titled 'Cleanup Policy'.

Name: A unique name for the cleanup policy. The value 'maven' is entered.

Format: The format that this cleanup policy can be applied to. The value 'maven2' is selected from a dropdown.

Notes: An empty text area for notes.

Criteria

- Published Before:** Restrict cleanup to components that were published to NXRM more than the given number of days ago. (Blob updated date)
A dropdown menu shows the value '30'.
- Last Downloaded Before:** Restrict cleanup to components that were last downloaded more than the given number of days ago. (Last downloaded date)
A dropdown menu shows the value '30'.
- Release Type:** Restrict cleanup to components that are of this release type
A dropdown menu shows the value 'Release Versions'.
- Asset Name Matcher:** Restrict cleanup to components which have at least one asset name matching the specified regular expression pattern.
Before using this feature refer to our [documentation](#) and preview results.
A dropdown menu shows the value 'org/sonatype/foo.*'.

At the bottom of the dialog are three buttons: 'Preview results', 'Create Cleanup Policy' (highlighted in blue), and 'Cancel'.

Before creation, but after you enter criteria, you can see a sampling of up to 50 components the criteria would delete by clicking the *Preview results* button.

Specifically, clicking will display a modal where you can select a repository, whose selection is limited by the format you had selected in creation, if applicable. On click of the *Preview* button the first 50 components alphabetically are returned along with a total number which would be cleaned below this. These results are sortable by column and filterable through the entire resultset, as opposed to just the first 50, in case you are trying to find a specific record or records.

After creation, a policy can then be associated with one or more repositories of the *Format* specified when creating or editing like repositories. See more about this in the [Cleanup Policy](#) section of [Managing Repositories and Repository Groups](#)(see page 0). Cleanup Policies are available for assignment to proxy and hosted repositories but not for group repositories.

Criteria

The available criteria for cleanup are shown in the table below as well as which format criteria are available on:

Criteria	Maven	Docker ¹	Helm	Yum	NPM	NUGET	Rubygems	p2	Pypi	Go	R	Raw	GitLFS	Bower	Appt	Cocoapods	Conda	Conan
Published Before	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓ ₂	✓	✓	✓	✓
Last Downloaded Before	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓ ₂	✓	✓	✓	✓
Release Type	✓	N/A		✓	✓													

Criter ia	M aven	D ocker ¹	H elm	Y um	N PM	N uG e t	R uby g e m s	p 2	P yPi	G o	R	R aw	G itL FS	B ower	A pt	C oco aP ods	C oonda	C oonan
Asset Name Matcher NEW IN 3.1.9	✓	✓	✓	N / A	✓	✓	✓	✓	✓	✓	✓	✓	✓	N / A	✓	✓	✓	✓

¹ - The Docker cleanup policy checks against the tagged components only. Only when the *Docker - Delete unused manifests and images* task has run will the cleanup be 'complete'.

² - Bower functionality is for Proxy repositories only.

- ⓘ "N/A" in the above table means Sonatype does not believe that the criteria applies to the format as built (though acknowledges it is possible people may have their own schemes). If you believe this to be untrue, feel free to contact us at nexus-feedback@sonatype.com⁶⁹¹ and let us know.

Published Before (Days)

This criteria has a number of days parameter to tell the policy how old content should be kept in days. As an example, if the criteria is set to 30 days then if no assets associated with a given component have been modified (including redeploy) within the last 30 days, this component will be soft deleted when the task executes the policy.

Last Downloaded Before (Days)

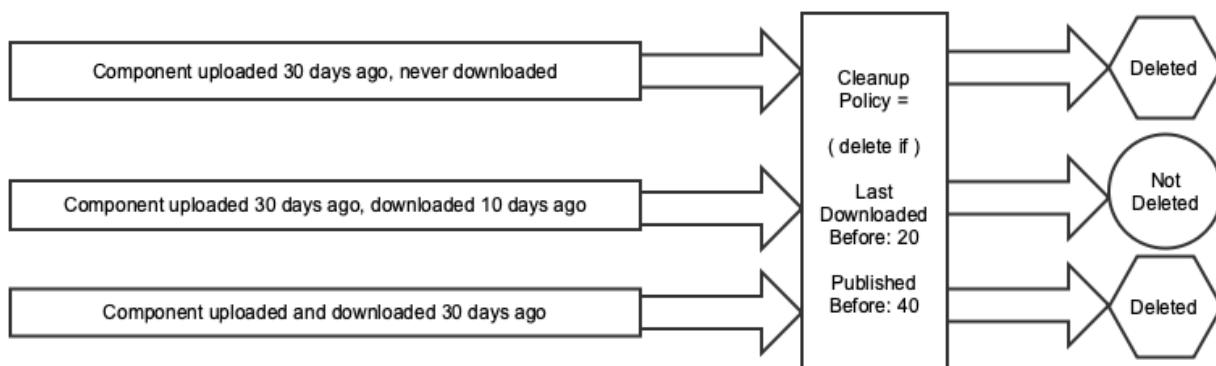
This criteria has a number of days parameter to tell the policy how old the last downloaded window of a component should be kept in days or which are never downloaded but updated more than the entered days(inclusive) before. As an example, if the criteria is set to 30 days then any component where none of the

⁶⁹¹ mailto:nexus-feedback@sonatype.com

assets associated with that component have been downloaded in the last 30 days or never downloaded and updated more than 30 days before(inclusive), this component will be soft deleted when the task executes the policy.

When we mark the component's last download date, we evaluate all the associated assets and use the last downloaded date from the most recently downloaded asset as the component's last downloaded date.

When this criteria is used with other policy options (like *Published Before* or *Asset Name Matcher*), then the items deleted will be the union between all the criteria. For Last Downloaded, as noted above, if never downloaded the fallback will be Published Date (not the *Published Before* policy) to assure all items are scanned for clean up. Here's a summary in diagram form:



In the first example in the diagram, the component was never downloaded so since the publish date was 30 days ago, it is effectively treated as downloaded 50 days ago and thus deleted by the policy of "Last Downloaded Before" being 20. It doesn't matter *Last Published* is 40 because it deletes the union of findings, i.e. if any condition is met, it deletes.

Release Type

This criteria has a configurable option to cleanup either prereleases or releases. It is worth noting that for repositories such as Maven this may only be relevant to repositories of mixed type. As an example, on a mixed Maven repository all snapshots would be cleaned up if this criteria was set to prerelease.

Prereleases are identified by different criteria for different formats:

Format	What is considered prerelease
Maven	The versions contains -SNAPSHOT
Npm	Npm uses semantic versioning ⁶⁹² so if the version contains "-" it is considered a prerelease.

⁶⁹² <https://semver.org/#spec-item-9>

Format	What is considered prerelease
Yum	The "release" property in the RPM header contains one of the following: <i>alpha, beta, rc, pre, prerelease, snapshot</i> <div style="border: 1px solid #f0e68c; padding: 5px; margin-top: 10px;">⚠ This is not case-sensitive</div>

Asset Name Matcher

- ⚠ Expressions are powerful and should be used with care. Always preview results to help avoid side effects.

Components in a blobstore consist of one or more assets (individual blobs). Asset names can be viewed when an asset is selected in the Browse or Search views.

The specified expression is evaluated against asset names. If there is a single match, the associated component and all of its contained assets (even those that did not match) are deleted.

An asset name does not begin with a leading slash. This is different than the asset request path value used when evaluating Content Selector (CSEL) or Routing Rule expressions.

Expression Syntax

The expression engine uses an [Elastic Search Regexp query syntax](#)⁶⁹³, from [Apache Lucene](#)⁶⁹⁴.

The expressions **are not** Perl (PCRE) or Java util.regex.Pattern compatible regular expressions and use a limited set of operators.

⁶⁹³ <https://www.elastic.co/guide/en/elasticsearch/reference/2.4/query-dsl-regexp-query.html#regexp-syntax>

⁶⁹⁴ http://lucene.apache.org/core/4_9_0/core/org/apache/lucene/util/automaton/RegExp.html

Expression Examples

Contents of target Repository	Policy Expression	Cleanup Case	Remaining items after Cleanup execution
<pre>hello/-/ hello-0.0.1.tgz</pre> <pre>hello/-/ hello-0.0.2.tgz</pre> <pre>hello/-/ hello-0.0.3.tgz</pre>	<pre>hello/-/hello-0.0. [2-9].tgz</pre>	All hello assets with point versions 2-9.	<pre>hello/-/ hello-0.0.1.tgz</pre>
<pre>org/example/ test.jar</pre> <pre>com/example/ test.jar</pre> <pre>test/example/ test.jar</pre>	(org com) / .*	Everything from the org and com groups.	<pre>test/example/ test.jar</pre>
<pre>org/sonatype/team1/ ui/5.0/ui-5.0.jar</pre> <pre>org/sonatype/team2/ format/1.0/ format-1.0.jar</pre> <pre>org/sonatype/team3/ database/10.0/ database-10.0.jar</pre>	<pre>org/sonatype/ team[2-3].*</pre>	Everything from the org.sonatype group on teams 2-3.	<pre>org/sonatype/team1/ ui/5.0/ui-5.0.jar</pre>

Contents of target Repository	Policy Expression	Cleanup Case	Remaining items after Cleanup execution
pool/main/libc/ libcap2/ libcap2_2.25-1.2_amd64.deb	pool/main/z/zsh/ zsh- common_5.4.2-3ubuntu3_all.deb	A specific single Apt asset.	pool/main/libc/ libcap2/ libcap2_2.25-1.2_amd64.deb
pool/main/z/zsh/ zsh_5.4.2-3ubuntu3_amd64.deb			pool/main/z/zsh/ zsh_5.4.2-3ubuntu3_amd64.deb
pool/main/z/zsh/ zsh- common_5.4.2-3ubuntu3_all.deb			

Cleanup Task

On start of a server which has cleanup abilities, a task named "Cleanup service" of type *Admin - Cleanup repositories using their associated policies* will be automatically created. By default, this task is scheduled to run daily at 1AM server time. Similar to other tasks, this task can be edited, disabled and executed manually if desired. If deleted, it will be automatically recreated on server restart. For more on tasks in general, see [Configuring and Executing Tasks](#)(see page 269).

When run, this task will execute cleanup of all the repositories which have a policy other than *None* set. There is no partial execution. This task cannot be manually created and either runs or does not.

General Guide (FAQ)

What Cleanup Policies are built in?

NXRM does not come with any cleanup policies built in. The assumption that you will want to define your own policies to your specifications and then assign them to the appropriate repositories, not that there is a one solution solves all environment or problems.

Is there anything more I need to do other than apply policies to free up space?

Yes, regardless of policy, no blob store space is reclaimed until you execute an *Admin - Compact blob store* task for blob stores of type *File* or the AWS Lifecycle runs for blob stores of type *S3*.

- ❗ Except in urgent situations where you need immediate space back, Sonatype recommends running the backup of your *File* type blob stores before you execute *Admin - Compact blob store* to make sure and have a copy of everything soft deleted by Cleanup.

How Docker Repositories Use Space

Docker repositories store and serve images. An **image** consist of docker tags, a manifest and layers.

A **manifest** describes what layers make up an image. Each manifest can reference many layers.

A docker **tag** is an alias pointing at a manifest. The 'latest' tag usually exists pointing at the latest published or proxied image.

A **layer** is unique and only stored once in a repository but can be referenced by many manifests.

Tags, manifests and layers are individual assets or blobs inside a blob store.

Blob store space is almost entirely consumed by docker layers. Space consumed by manifests and tags is negligible by comparison.

Finally, Docker V2 API implements [resumable uploads](#)⁶⁹⁵. Docker uploads can be abandoned through normal use and this can leave large files dangling in temporary blob store files.

How much space does a Docker Image consume inside a blobstore?

Each image part is stored as an individual asset blob inside NXRM - each one of these has its own individual part size. If you select any one part (blob, manifest, tag) inside NXRM UI, the asset *Summary → File size* section shows the individual asset size as NXRM stores it in the blobstore.

An image manifest asset lists each layer size inside the manifest file. Even if you add up all the sizes listed in the manifests, this total size may still not match what `docker image ls` prints after an image is docker pulled, depending on the operating system where the command is run. This is normal.

A crude method to understand the total physical size of all manifest layers is to download the image manifest file, add up all the individual sizes for the layers inside the manifest. This will roughly total to amount of disk space needed to store that single image on the NXRM side. However, since images share layers, it would not be accurate to simply add all of these image manifest totals up and conclude that is how much storage a NXRM blobstore needs. The space used could overlap shared layers of other images.

Docker Cleanup Strategies

The following table outlines what tasks will soft-delete blobs in blob stores for Docker repositories:

⁶⁹⁵ <https://docs.docker.com/registry/spec/api/#initiate-blob-upload>

Optimal Run Order	Task Type	Feature	What it deletes?
1	Docker - Delete incomplete uploads	Tasks	soft-delete dangling uploads in temporary blob store storage which have not been resumed
2	Admin - Cleanup repositories using their associated policies	Cleanup Policies	soft-delete old published or downloaded docker components i.e. tags, not layers or manifests
3	Docker - Delete unused manifests and images	Tasks	soft-delete orphaned layers and manifests no longer referenced by tags, possibly orphaned by cleanup policies

The following table outlines what features actually perform a hard-delete (i.e. free storage space) of soft-deleted blobs:

Feature	Blob Store Type
Admin - Compact blob store task	File
S3 blob store configuration Expiration Days	AWS S3

How can I tell what repositories are using the most space?

Refer to the Support article: [Investigating Blobstore and Repository Size and Space Usage](#)⁶⁹⁶, and the `nx-blob-repo-space-report.groovy` script that is provided.

For purposes of repository size, you'll want to look at `totalBytes` within the output.

 The above script may have performance issues with large blobstores.

How do Cleanup Policies impact other "delete" tasks?

The implementation documented on this page should replace the need for any `Maven - Delete usused SNAPSHOT` and `Repository - Delete unused components` tasks by using the `Last Downloaded Before` criteria.

`Maven - Delete SNAPSHOT` tasks are not yet completely replaced .

As mentioned [above\(see page 0\)](#), `Docker - Delete unused manifests and images` is not replaced. In fact, it is necessary to run after your cleanup policies to remove orphaned layers and manifests.

⁶⁹⁶ <https://support.sonatype.com/hc/en-us/articles/115009519847>

Docker - Delete Incomplete Uploads, Admin - Cleanup tags and any other task specific to delete or cleanup not mentioned here is not covered by policy cleanup at all and need to continue to be used as they are. See [here](#)⁶⁹⁷ for more about these tasks in general.

How do I upgrade existing tasks?

There is no migration in place, so creation of similar policies then assignment to repositories as well as deletion or disabling of existing tasks must all be done manually. Because cleanup is also implemented as a scheduled task, there is no collision if both remain running however it is a resource drain. We recommend rollover to this feature once configured and comfortable.

What is the needed security to use this feature?

Only users with the admin (i.e.

`nexus:*`) privilege can use the *Cleanup Policy* left navigation item. Any user with privilege to edit a repository (such as `nexus:repository-admin:maven2:maven-central:edit` for the default maven-central repository) can adjust the policy a repository uses. Permission to edit the cleanup task is covered by the same permissions as other tasks (`nx-tasks-update` or `nx-tasks-all`). There is no individual privilege for just the Cleanup Task.

What if I'm on an older version of NXRM3?

If you're on an older version of NXRM3, you'll need to use the established delete tasks in your version. See [How do Cleanup Policies impact other "delete" tasks?\(see page 306\)](#) FAQ question for more about those tasks.

Any other tips?

See the [Keeping Disk Usage Low\(see page 307\)](#) subpage for further tips.

Keeping Disk Usage Low

Generally, we recommend using [Cleanup Policies\(see page 297\)](#) and the *Admin - Compact blob store* task to keep disk usage low. However, this page details several other items for when policies are not enough .

⁶⁹⁷<https://help.sonatype.com/display/NXRM3M/System+Configuration#SystemConfiguration-TypesofTasksandWhentoUseThem>

Tasks

If Cleanup Policies are not meeting your needs, you may need to use the existing tasks. See [the FAQ section of Cleanup Policies\(see page 0\)](#) for more about what Cleanup Policies cover and what needs Tasks used instead.

Task Reminder

Like Cleanup, these tasks soft delete and the *Admin - Compact blob store* task is needed after to actually free your disk space.

For More Complex Cleanup Scenarios

For scenarios the policies and tasks don't cover with their provided variables, we suggest using the [REST API\(see page 464\)](#) to find and delete what you need via REST call or Scripting.

For example, if you'd like to delete all components named alpha, you can perform the following curl to identify them:

```
curl -X GET "http://localhost:8081/service/rest/v1/search?name=alpha" -H "accept: application/json"
```

Depending on your results, you can then call individual deletes or design a script to remove them all. Example of what a delete curl might look like:

```
curl -X DELETE "http://localhost:8081/service/rest/v1/components/bWF2ZW4tcmVsZWFzXm60Dg00TFjZDFkMTg1ZGQzMzgwYjA3YWQzNDAwOGVmNTc" -H "accept: application/json"
```

Examining Blobstore Space Usage

If you would like more insight into how blobstore space is being consumed to make better informed configuration decisions, you can run groovy scripts either from the command line or from within NXRM using the *Execute Script* task. This [Knowledge Base article](#)⁶⁹⁸ provides samples to get you started.

⁶⁹⁸ <https://support.sonatype.com/hc/en-us/articles/115009519847-Investigating-Blobstore-Space-Usage>

Totally Out of Space / Seeing Errors?

If you are currently out of disk space, there are a couple of other Knowledge Base articles you can look at to free up space urgently and get the system up and running:

[What to Do When the Database is Out of Disk Space](#)⁶⁹⁹

[What to Do When the Blobstore is Out of Disk Space](#)⁷⁰⁰

Security

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 gives you privilege to all aspects of the NXRM system and uses the username **admin** and the initial password can be found in an `admin.password` file in the `$data-dir` directory after startup.

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 313), [Roles](#)(see page 317) and [Users](#)(see page 320)
- selection of security backend systems called [Realms](#)(see page 310) 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 338)

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.

⁶⁹⁹ <https://support.sonatype.com/hc/en-us/articles/360000052388>

⁷⁰⁰ <https://support.sonatype.com/hc/en-us/articles/360000096228>

The role-based access control system is backed by different authentication and authorizations systems as documented in [Realms\(see page 310\)](#) 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\(see page 310\)](#)
- [Privileges\(see page 313\)](#)
- [Roles\(see page 317\)](#)
- [Users\(see page 320\)](#)
- [Anonymous Access\(see page 323\)](#)
- [LDAP\(see page 324\)](#)
- [SAML\(see page 331\)](#)
- [Security Setup with User Tokens\(see page 335\)](#)
- [Authentication via Remote User Token\(see page 337\)](#)
- [Configuring SSL\(see page 338\)](#)
- [Default Role\(see page 346\)](#)

Realms

 Available in Nexus Repository OSS and Nexus Repository Pro

The feature view for security realms administration displayed in *Figure: "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.

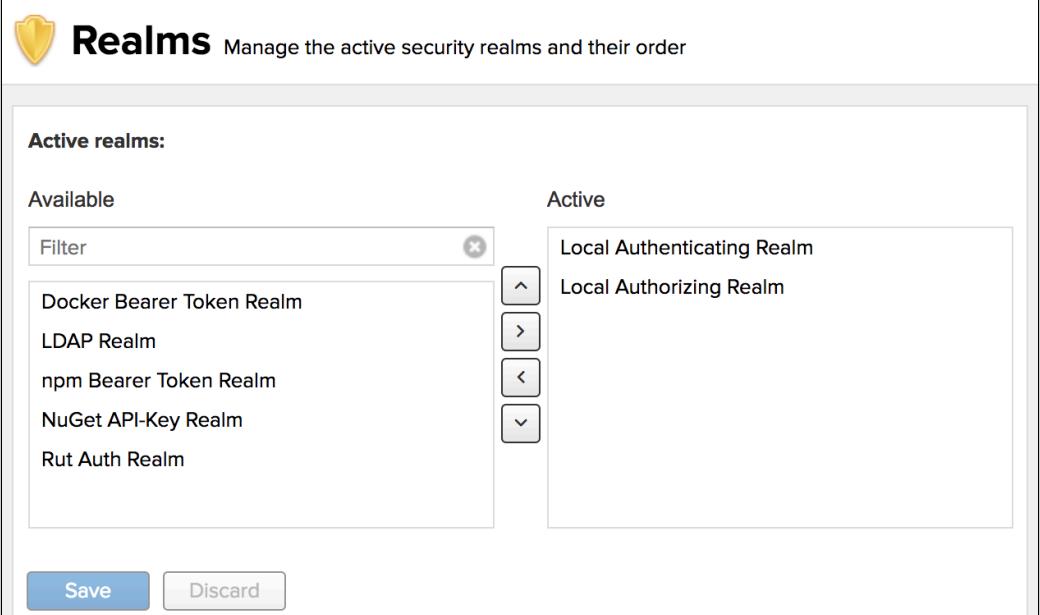


Figure: Security Realms Administration

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.

Recommended Ordering

Sonatype recommends keeping the Local realms at the top of the active list. In the event of system recovery, if you have them lower in the order (or removed), restoration may be more difficult.

Crowd Realm

This realm identifies external storage in an Atlassian Crowd system with details documented in [Atlassian n Crowd Support](#)(see page 347).

Default Role Realm

This realm will append the configured role to all users when they are authenticated, see [the Default Role page](#).⁷⁰¹

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 [Authentication\(see page 394\)](#), found in [the Docker section](#)⁷⁰².

LDAP Realm

This realm identifies external storage in an LDAP system including e.g., Microsoft ActiveDirectory, ApacheDS, OpenLDAP with details documented in [LDAP\(see page 324\)](#).

npm Bearer Token Realm

This realm permits users with previously generated bearer tokens to publish *npm* packages. See [npm Security\(see page 425\)](#) 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 [NuGet Repositories\(see page 428\)](#).

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\(see page 337\)](#).

SAML Realm

This realm uses an external Identity Provider (IdP) to handle authentication, details documented in [the SAML page](#)⁷⁰³.

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\(see page 336\)](#).

- ⚠ 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.

⁷⁰¹ <https://help.sonatype.com/display/NXRM3M/Default+Role>

⁷⁰² <https://help.sonatype.com/display/NXRM3/Private+Registry+for+Docker#PrivateRegistryforDocker-AnonymousReadAccess>

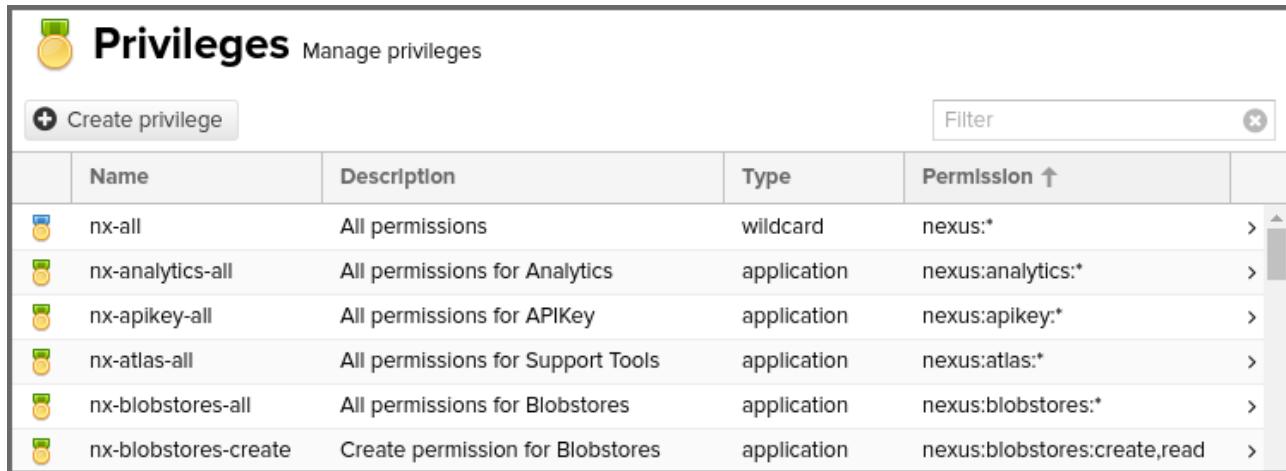
⁷⁰³ <https://help.sonatype.com/display/NXRM3M/SAML>

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: "Partial List of Security Privileges"*. This feature allows you inspect existing privileges and create custom privileges.



	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: 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

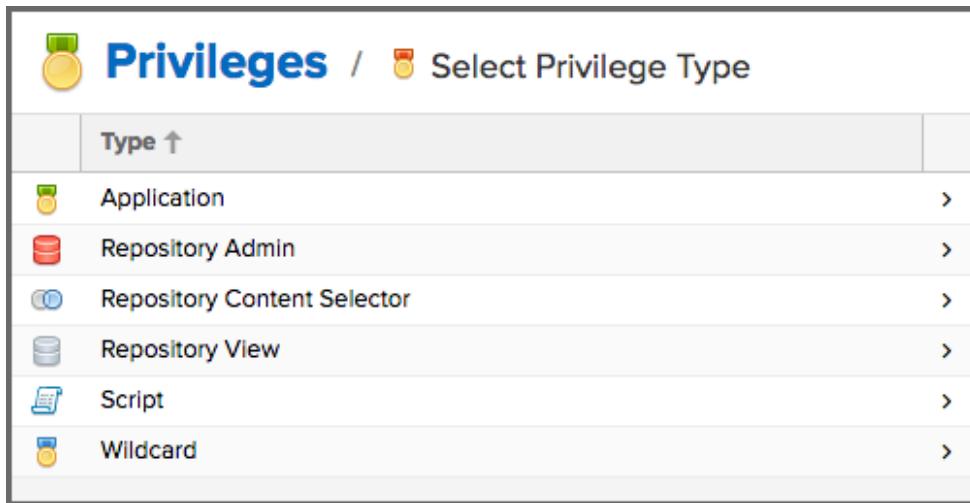
Permission

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: “Choosing Privilege Types”*.



The screenshot shows a list of privilege types under the heading "Privileges / Select Privilege Type". The list includes:

Type ↑
Application
Repository Admin
Repository Content Selector
Repository View
Script
Wildcard

Figure: 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](#)(see page 234)

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](#)(see page 464)

Wildcard

These are privileges that use patterns to group other privileges

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.

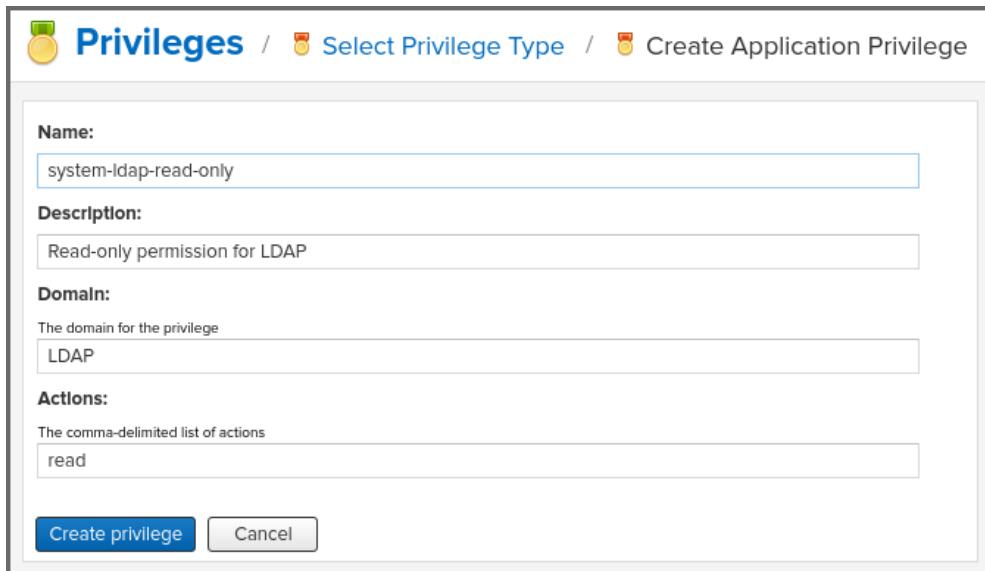
*

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-based form titled "Create Application Privilege". The form has the following fields and values:

- Name:** system-ldap-read-only
- Description:** Read-only permission for LDAP
- Domain:** LDAP
- Actions:** read

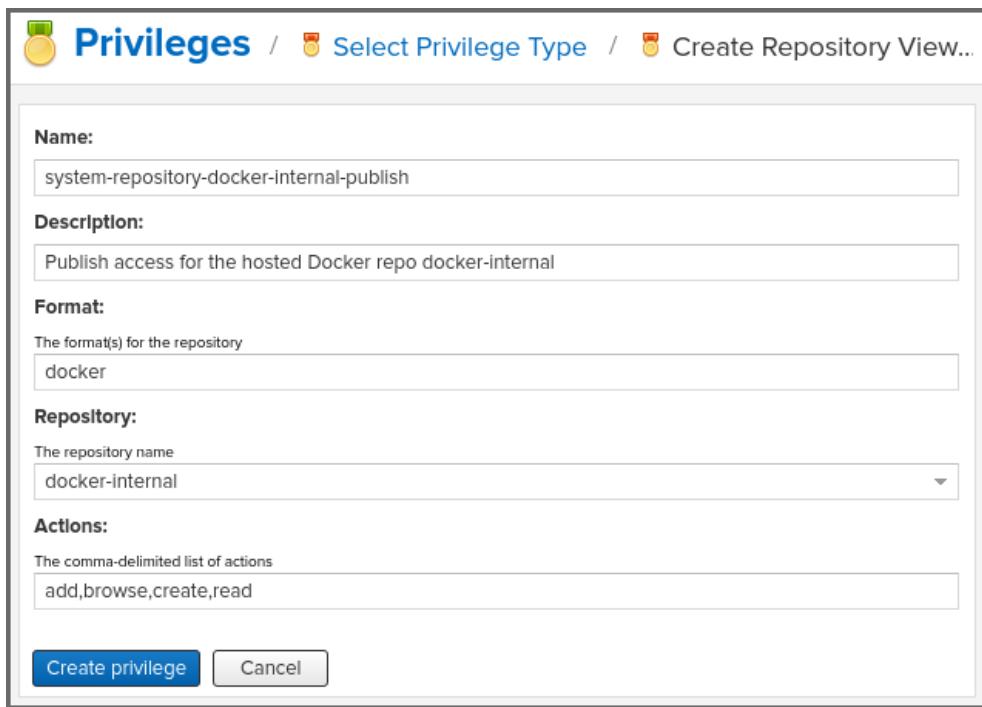
At the bottom of the form are two buttons: "Create privilege" (highlighted in blue) and "Cancel".

Figure: Creating an Application Privilege

The form provides *Name*, *Description*, *Domain*, and *Actions*. In *Figure: "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:



The form provides *Name*, *Description*, *Format*, *Repository*, and *Actions*. In Figure: "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.

Figure: Creating a Repository View Privilege

The form provides *Name*, *Description*, *Format*, *Repository*, and *Actions*. In Figure: "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.

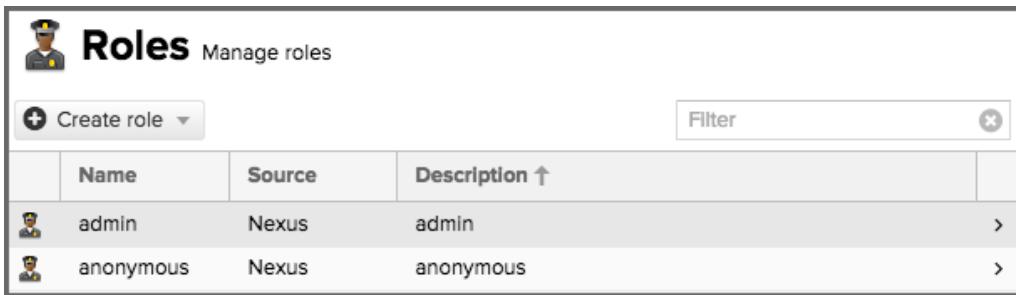
Roles

 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: "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.

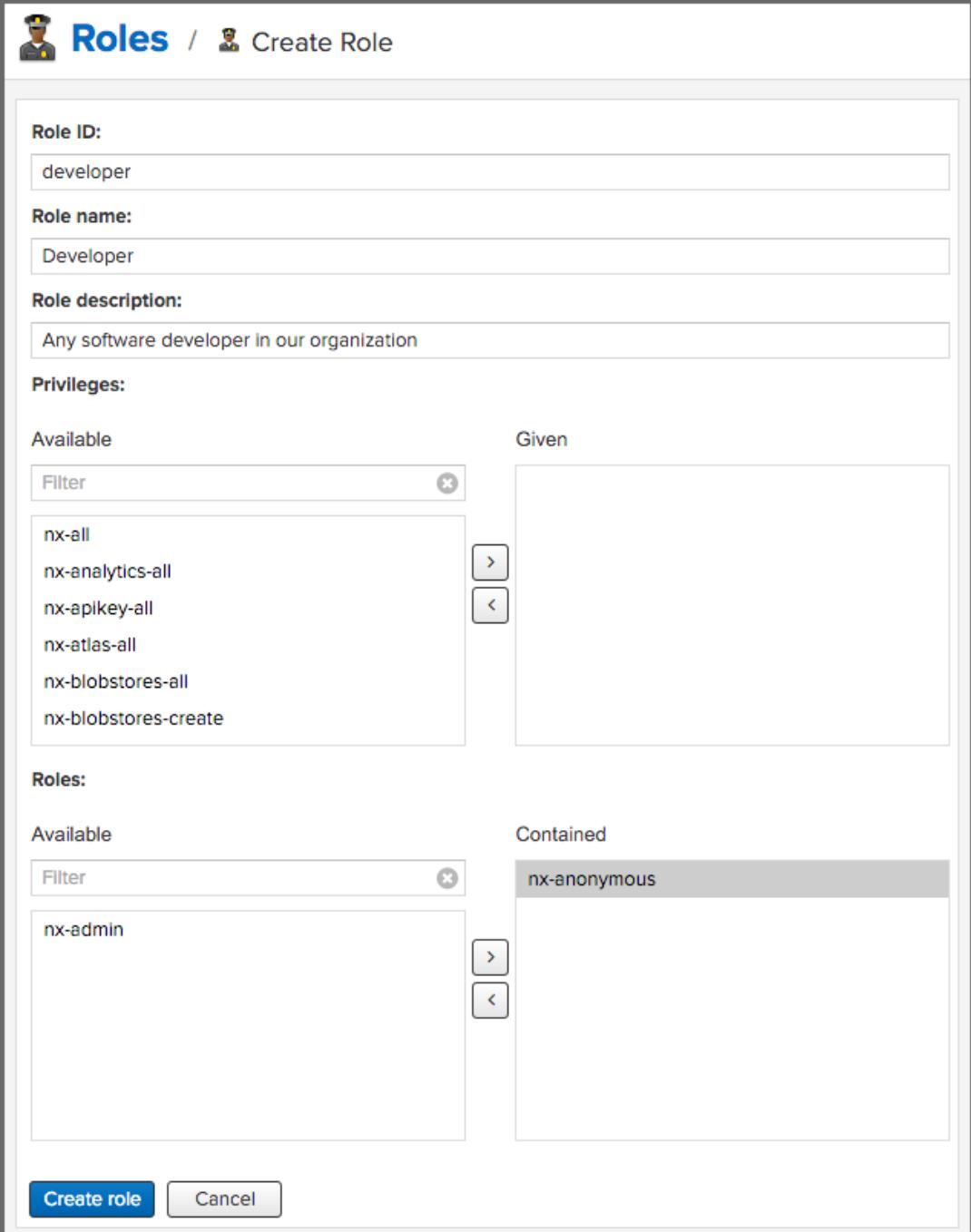


The screenshot shows the 'Roles' management interface. At the top, there's a header with a user icon and the word 'Roles'. Below it is a 'Create role' button with a dropdown arrow and a 'Filter' input field with a clear button. A table lists two roles: 'admin' and 'anonymous'. The table has columns for Name, Source, and Description. Each row has a small user icon in the first column and a right-pointing arrow in the last column.

	Name	Source	Description ↑	
	admin	Nexus	admin	>
	anonymous	Nexus	anonymous	>

Figure: 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: “Creating a New Role”*.



Role ID:
developer

Role name:
Developer

Role description:
Any software developer in our organization

Privileges:

Available	Given
Filter nx-all nx-analytics-all nx-apikey-all nx-atlas-all nx-blobstores-all nx-blobstores-create	

Roles:

Available	Contained
Filter nx-admin	nx-anonymous

Create role **Cancel**

Figure: 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: "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*.

- ⓘ 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.

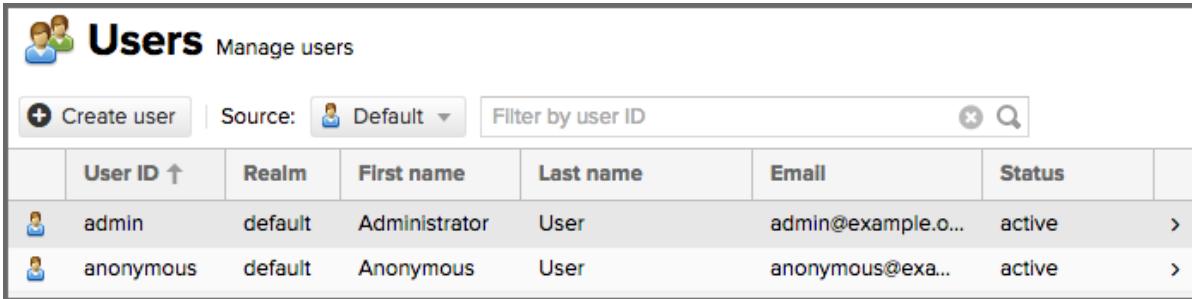
Any user that is part of the `scm` group in *LDAP*, receives all the privileges defined in the created role allowing you to adapt your generic role in *LDAP* to the repository manager-specific use cases you want these users to be allowed to perform.

Users

- ⓘ 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 initial password for the *admin* user can be found in an `admin.password` file found in the `$data-dir` directory after starting the server. The *Users* feature view displayed in *Figure: "Feature View with List of Users"* can be accessed via the *Users* item in the

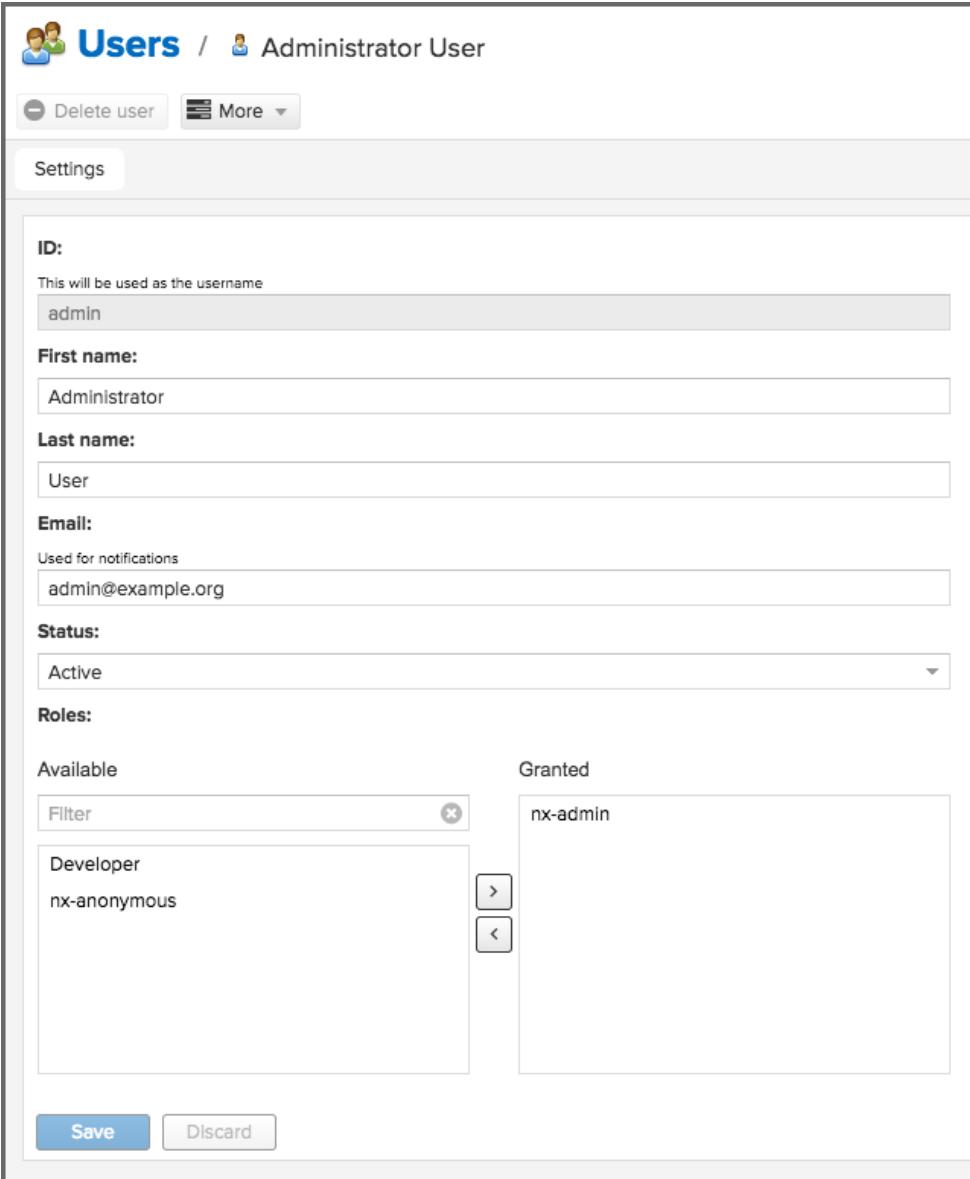
Security section of the *Administration* menu. The list shows the users *User ID*, *First Name*, *Last Name* and *Email* as well as what security *Realm* is selected and if the accounts *Status* is *active* or *disabled*. The *Default* security realm is the local NXRM realm.



	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: 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: “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' page in the Nexus Repository Manager 3 interface. The title bar says 'Users / Administrator User'. There are buttons for 'Delete user' and 'More'. A 'Settings' tab is selected. The form fields are as follows:

- ID:** admin (This will be used as the username)
- First name:** Administrator
- Last name:** User
- Email:** admin@example.org (Used for notifications)
- Status:** Active
- Roles:**
 - Available:** Developer, nx-anonymous
 - Granted:** nx-admin

At the bottom are 'Save' and 'Discard' buttons.

Figure: 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 317).

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.

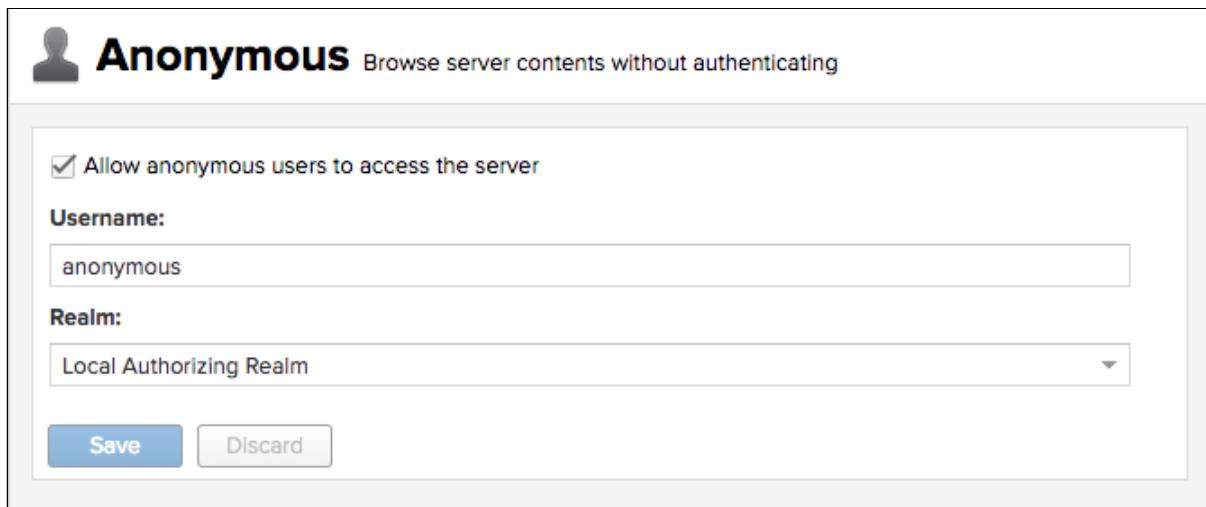
⚠ Ensure to change the password of the admin user to avoid security issues. Alternatively create other users with administrative rights and disable the default admin user.

Anonymous Access

i Available in Nexus Repository OSS and Nexus Repository Pro

If Anonymous Access is enabled, 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: “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\(see page 320\)](#) feature view.



The screenshot shows the 'Anonymous' configuration page. At the top, there is a header with a user icon and the text 'Anonymous'. Below the header, there is a sub-header 'Browse server contents without authenticating'. A checkbox labeled 'Allow anonymous users to access the server' is checked. Underneath the checkbox, there are two input fields: 'Username' containing 'anonymous' and 'Realm' containing 'Local Authorizing Realm'. At the bottom of the form are two buttons: 'Save' and 'Discard'.

Figure: 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

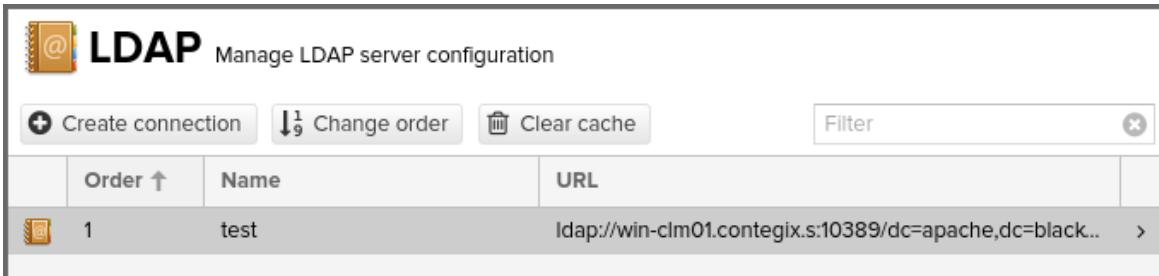
Activate your *LDAP Realm* by following these steps:

- 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: “LDAP Feature View”*, is available via the *LDAP* item in the *Security* section of the *Administration* main menu.



The screenshot shows the 'LDAP' feature view with the title 'Manage LDAP server configuration'. It includes a toolbar with 'Create connection', 'Change order', 'Clear cache', and a 'Filter' input field. A table lists one configuration entry:

Order ↑	Name	URL
1	test	ldap://win-clm01.contegix.s:10389/dc=apache,dc=black...

Figure: 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 DN

This field further qualifies the connection to the LDAP server. It usually corresponds to the domain name of an organization. For example: `dc=example,dc=com`.

Note: If the values in your Search base DN 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](#)⁷⁰⁴.

For a full discussion of LDAP authentication approaches, see RFC-2829 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.

Password

⁷⁰⁴ <http://www.faqs.org/rfcs/rfc2195.html>

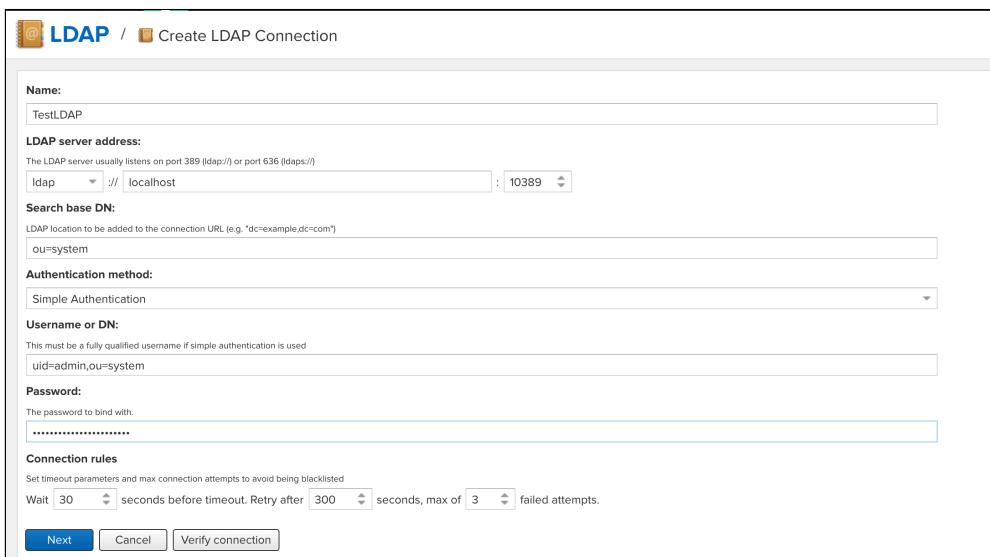
Password for the *Username or DN* configured above.

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: “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.



The screenshot shows the 'Create LDAP Connection' configuration page. The fields filled in are:

- Name:** TestLDAP
- LDAP server address:** ldap://localhost:10389
- Search base DN:** ou=system
- Authentication method:** Simple Authentication
- Username or DN:** uid=admin,ou=system
- Password:** (Redacted)
- 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.

At the bottom are buttons for **Next**, **Cancel**, and **Verify connection**.

Figure: 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: “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.

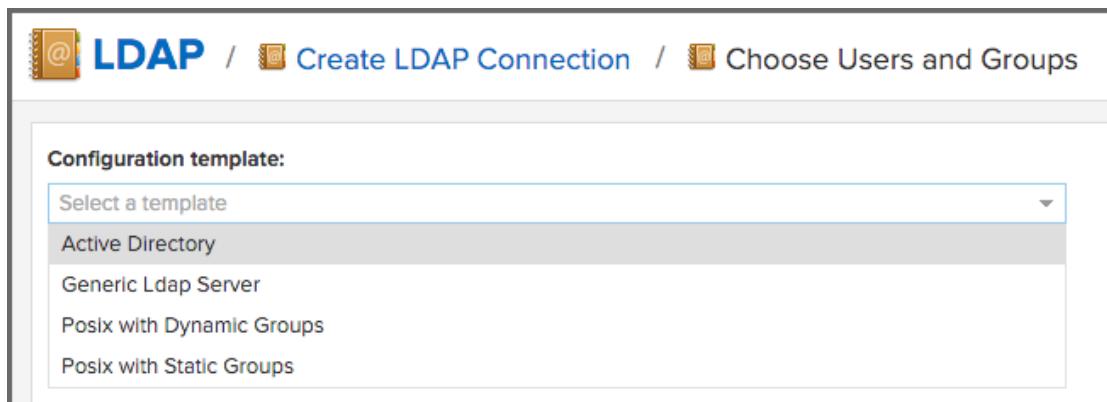


Figure: Configuration Template for Users and Groups

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

User relative DN

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

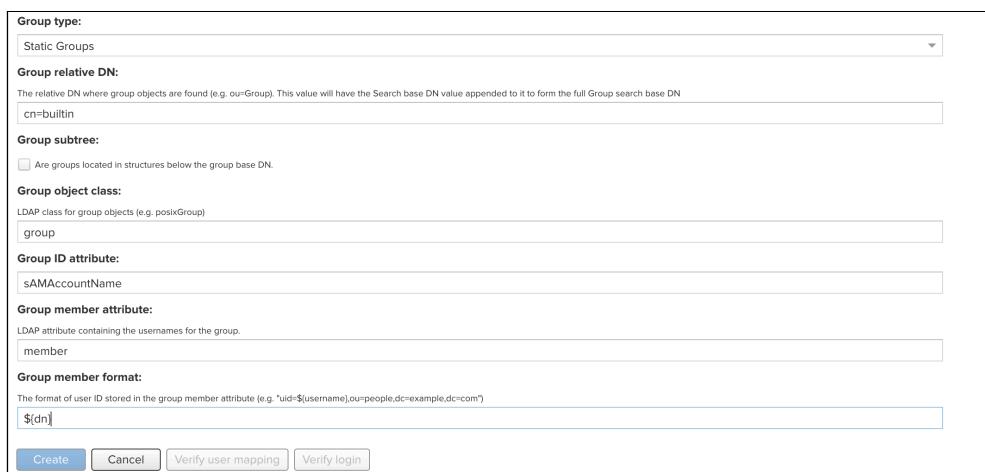
This is the attribute of the Object class that supplies the email address of the user. The repository manager uses this attribute for the *Email attribute* of the user. It is used for email notifications of the user.

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.



The screenshot shows a configuration dialog for a static group. The 'Group type:' dropdown is set to 'Static Groups'. The 'Group relative DN:' input field contains 'cn= builtin'. The 'Group subtree:' checkbox is unchecked. The 'Group object class:' input field contains 'group'. The 'Group ID attribute:' input field contains 'sAMAccountName'. The 'Group member attribute:' input field contains 'member'. The 'Group member format:' input field contains '\${dn}'. At the bottom are buttons for 'Create', 'Cancel', 'Verify user mapping', and 'Verify login'.

Figure: Static Group Element Mapping

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

Group relative DN

This field is similar to the *User Relative 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 Group relative DN, set the field to false. If a group can be defined in a tree of organizational units under the Group relative 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.

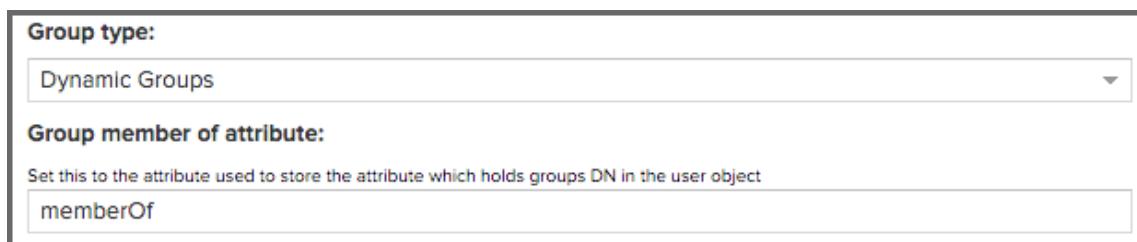
Group member attribute

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.



Group type:
Dynamic Groups

Group member of attribute:
Set this to the attribute used to store the attribute which holds groups DN in the user object
memberOf

Figure: 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: "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\(see page 317\)](#) and [Users\(see page 320\)](#).

SAML

You can configure NXRM3 to work with a SAML Identity Provider for authentication via Single Sign-On (SSO) and to send user groups to NXRM3 for authorization.

 **Available in Nexus Repository Manager Pro only**

NXRM3 SAML Support

NXRM3 implements the Web Browser SSO Profile from the [SAML 2.0 specification⁷⁰⁵](#). Supported bindings for sign-on are HTTP-POST (the default) and HTTP-Redirect for requests to the Identity Provider and HTTP-POST binding for responses from the Identity Provider. The Basic Attribute Profile is used for retrieving further information regarding the subjects and supported attributes are explained in [SAML Configuration Details\(see page 333\)](#).

 SAML provides access to the NXRM3 UI. For clients that don't support SSO, NXRM3 can generate [user tokens\(see page 335\)](#), which can be used instead of a user name and password for basic authentication or to make REST API calls.

Integrating NXRM3 with an Identity Provider

Note that configuring SAML in NXRM3 requires the nx-all privilege in NXRM3.

Configuring SAML can be achieved in a few simple steps:

⁷⁰⁵ <http://docs.oasis-open.org/security/saml/v2.0/>

1. Download the Identity Provider's [SAML Metadata](#)⁷⁰⁶ file.
2. In NXRM3, upload the identity provider metadata as described in the [section below](#)(see page 332).
3. Download NXRM3's SAML Metadata (also known as the Service Provider Metadata) to provide to the Identity Provider. Note that some Identity Providers may be configured differently. It can be downloaded with the SAML REST API:

```
curl -u admin:admin123 -X GET "http://localhost:8081/service/rest/v1/security/saml/metadata"
```

4. If applicable, upload NXRM3's SAML Metadata (downloaded in the previous step) to the Identity Provider.
 - This is also referred to as registering a Service Provider to an Identity Provider.
 - If the Identity Provider does not allow uploading NXRM3's SAML Metadata, then you can inspect it to extract the needed values. Also note that the "Audience" of your Identity Provider should be NXRM3's Entity ID, and its "Recipient" should be NXRM3's assertion consumer service URL. Please note that NXRM3's assertion consumer service URL is /saml.
5. Configure the Identity Provider to send expected subject information as basic attributes. Note that these attribute names can be customized later.
6. Enable the *SAML Realm* in NXRM3. See the [section below](#)(see page 333).
7. Enable *User Tokens* in NXRM3. See the [section below](#)(see page 333).

At this point, integration is done and Single Sign-On using SAML should be functional via the [SAML Workflow](#)(see page 334).

Note that:

- As long as a SAML configuration exists in NXRM3 and the SAML Realm is enabled, the user will be prompted to *Sign in with SSO*, or *Sign In Without SSO* for signing in using non-SAML realms.
- If no SAML configuration exists or the SAML Realm is not enabled, the Single Sign-On prompt dialog will not appear.
- When you delete an existing SAML configuration and create a new one, even if you use the same Identity Provider Metadata XML, NXRM3 will generate new keys, requiring SAML Metadata XML to be downloaded and re-registered to the Identity Provider.

Uploading the Identity Provider Metadata and Configuring the Attribute Names Using the UI

- Access the SAML Configuration UI using the *SAML* item of the *Security* administration section.
- When SAML is not configured in NXRM3, the *SAML Identity Provider Metadata XML* will be empty, and the attributes will have their default values populated.
- Upload the Identity Provider's Metadata XML by pasting the XML directly in the text area.
- Modify the response/assertion signature validation if needed.

⁷⁰⁶ <http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf>

- Modify the Attribute Names if needed.
- Press Save.

Enabling the SAML Authentication Realm

Activate your *SAML Realm* by following these steps:

- Navigate to *Realms* in the *Security* administration section
- Select the *SAML Realm* and add it to the list of *Active realms* on the right
- Ensure that the *SAML 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 SAML authentication offline or unavailable. When users choose *Sign In Without SSO*, the other realms will be used for authentication, in the order they appear.

Enabling User Tokens

Activate *User Tokens* by the following steps:

- Navigate to *User Tokens* in the *Security* administration section.
- Click *Enable User Tokens*.
- Press Save.

The *User Token Realm* will automatically be added to the active Realms.

Refer to the section on [User Tokens\(see page 335\)](#) for more details.

SAML Configuration Details

Entity ID

Entity ID is a unique identifier that will be used when registering NXRM3 to the Identity Provider as a Service Provider. It must be a valid URI. The default value <BaseUrl>/service/rest/v1/security/saml/metadata is recommended and normally there is no need to modify the Entity ID, but it is required if the deployed instance does not have a fixed URL or port.

Attribute Names

NXRM3 expects the following basic SAML attributes to carry/extract the user information:

- *username* attribute
- *first name* attribute

- last name attribute
- e-mail attribute
- groups attribute; SAML groups will be mapped to NXRM roles

These attribute names must be configured on the Identity Provider, and must match the SAML Configuration in NXRM3. Note that the values are case-sensitive.

For example, if the Identity Provider sends the email information of a user with the attribute `user-email-address`, please make sure you configure the email attribute name to `user-email-address` in NXRM3 as well.

All of these properties are optional to configure. If they are not explicitly configured, default values for them will be used.

Securing SAML Integration

Signed requests and responses between the Identity Provider and NXRM are supported and recommended for security purposes. Signing responses can only be configured from the Identity Provider and not within NXRM3. In case signed messages are used and the Identity Provider's signature keys are renewed (or changed for any other reason), Identity Provider metadata in NXRM3's SAML configuration must be updated.

NXRM3's SAML configuration can be set to validate the Identity Provider's signature(s) on its response and/or its assertions within that response by setting

- the validate response signature
- the validate assertion signature

options. By default, if a signing key is found in the Identity Provider metadata, then NXRM3 will attempt to validate signatures on the response and its assertions.

User SAML Workflow

Once NXRM3 is configured to use SAML, users can log into the NXRM3 UI.

- Login can be done using the Identity Provider's login experience by clicking the Single Sign-On button at the user login.
- Once the user logs in to the Identity Provider, they will be redirected back to NXRM3 with an active session.
- Logging out of NXRM3 will not invalidate the session on the Identity Provider.
- The user can generate a [user token](#)(see page 335) to use for secure authentication with format clients.

Some additional considerations:

- Format clients like mvn, docker and npm don't support SSO, so users that authenticate with SAML must use user tokens.

- Any existing sessions will not be invalidated once SAML is configured meaning any users already logged in when SAML is configured will need to logout and login again via the Single Sign-On button to use SAML SSO.
- Roles can be assigned to SAML users by an NXRM3 administrator if additional role based security is required.

After a user has logged into NXRM3, they can obtain a user token:

- The user must access the menu by selecting their username on the top right area of the main toolbar and select *User Token*.
- The user must then click *Access User Token*.
- The token can then be copied into the configuration file for the format client.

Revoking User Access

User access is revoked by

- Disabling the user in the Identity Provider.
- Revoking user tokens as an NXRM3 administrator.

To revoke user tokens, the User Token REST API can be invoked:

```
curl -u admin:admin123 -X DELETE "http://localhost:8081/service/rest/beta/security/user-tokens"
```

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 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 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+ZBiT1F:76xSi+HAQvYHZH8kgJldWD7aJnPgCrHG/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 Rut 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](#)⁷⁰⁷. In many cases, this is achieved via a server like [Apache HTTPD](#)⁷⁰⁸ or [nginx](#)⁷⁰⁹ proxying the repository manager. These servers can in turn defer to other authentication storage systems e.g., via the [Kerberos](#)⁷¹⁰ 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](#)(see page 259) 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 310).

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.

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](#)(see page 234)
- SSL/TLS secured servers - e.g. for SMTP/email integration documented in [Email Server](#)(see page 261)
- LDAP servers configured to use LDAPS
- specialized authentication realms such as the [Crowd](#)(see page 347) realm.

⁷⁰⁷ <http://shibboleth.net/>

⁷⁰⁸ <http://httpd.apache.org/>

⁷⁰⁹ <http://nginx.org/>

⁷¹⁰ <http://web.mit.edu/kerberos/>

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](#)(see page 240) 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: "Certificate Details Dialog to Add an SSL to the Nexus Truststore"*.

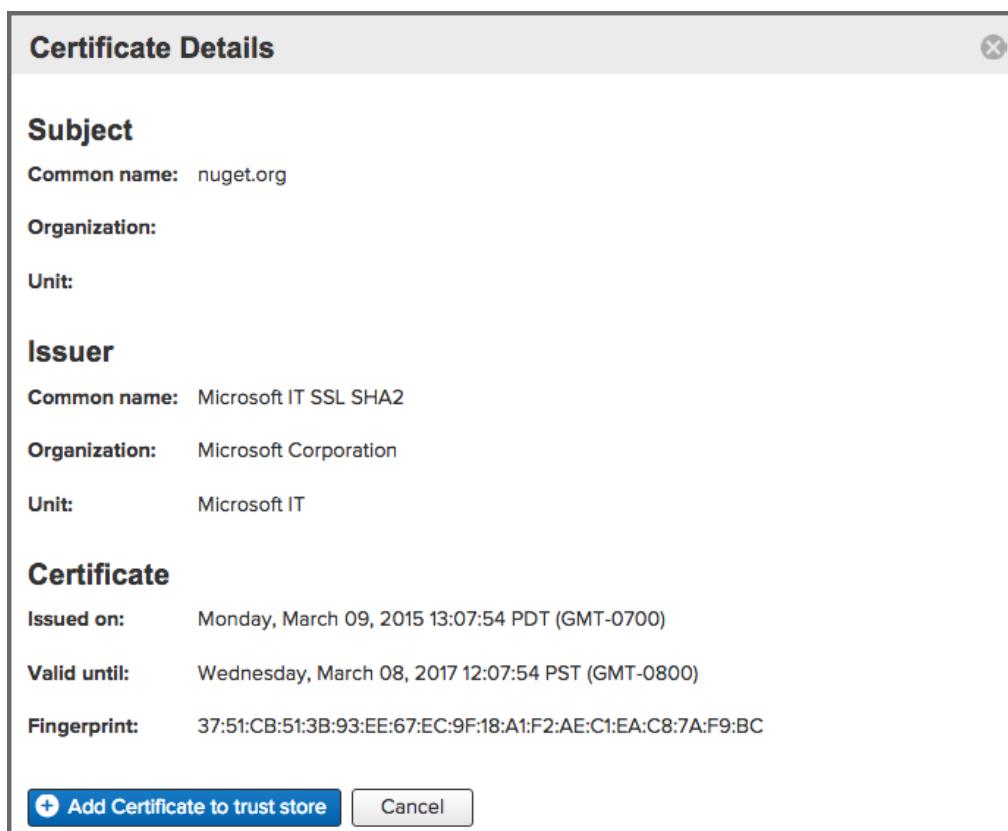


Figure: Certificate Details Dialog to Add an SSL to the Nexus Truststore

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 340). 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.

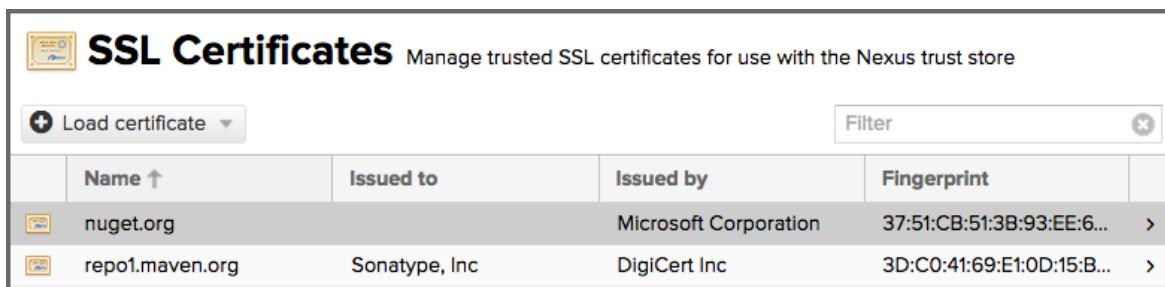
-  When removing a remote trusted certificate from the truststore, a repository manager restart is required before a repository may become untrusted.

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.



The screenshot shows the 'SSL Certificates' administration interface. At the top, there's a header with the title 'SSL Certificates' and a sub-instruction 'Manage trusted SSL certificates for use with the Nexus trust store'. Below the header is a search bar labeled 'Filter' with a clear button. Underneath is a table with four columns: 'Name ↑', 'Issued to', 'Issued by', and 'Fingerprint'. There are two rows of data:

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: 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: “Providing a Certificate in PEM Format”*.

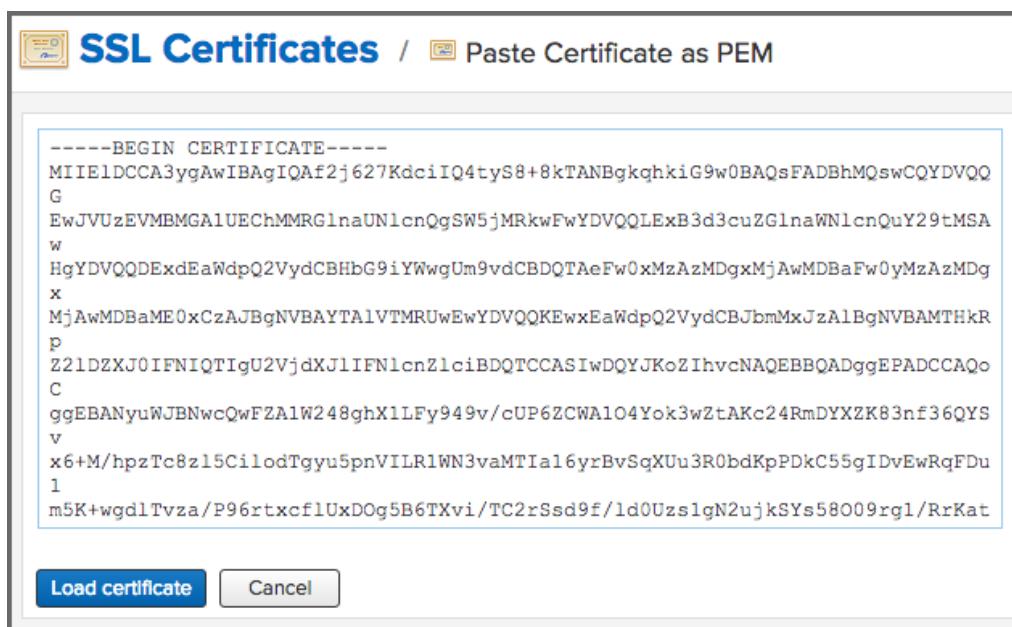
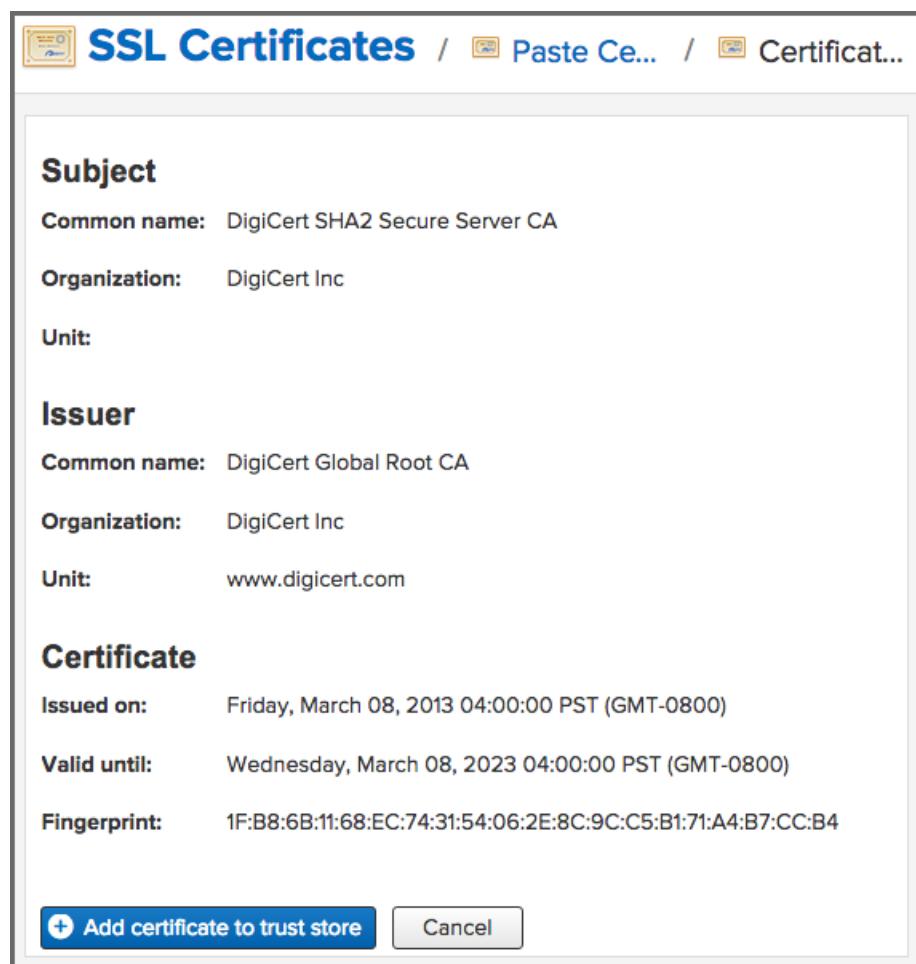


Figure: Providing a Certificate in PEM Format

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: "Certificate Details Displayed after Successful Retrieval or Parsing"*. Please review the displayed information carefully before clicking *Add Certificate* to establish the truststore addition.



The screenshot shows the 'SSL Certificates' page in the Nexus Repository Manager. The top navigation bar includes icons for 'SSL Certificates', 'Paste Ce...', and 'Certificat...'. The main content area is divided into sections: 'Subject', 'Issuer', and 'Certificate'.
Subject:
Common name: DigiCert SHA2 Secure Server CA
Organization: DigiCert Inc
Unit:
Issuer:
Common name: DigiCert Global Root CA
Organization: DigiCert Inc
Unit: www.digicert.com
Certificate:
Issued on: Friday, March 08, 2013 04:00:00 PST (GMT-0800)
Valid until: Wednesday, March 08, 2023 04:00:00 PST (GMT-0800)
Fingerprint: 1F:B8:6B:11:68:EC:74:31:54:06:2E:8C:9C:C5:B1:71:A4:B7:CC:B4
At the bottom are two buttons: a blue 'Add certificate to trust store' button with a plus icon, and a white 'Cancel' button.

Figure: 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`⁷¹¹

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:

⁷¹¹ <http://docs.oracle.com/javase/8/docs/technotes/tools/index.html#security>

- a basic understanding of SSL certificate technology and how the Java VM implements this feature
- 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](#)⁷¹².

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

⁷¹² <https://support.sonatype.com/hc/en-us/articles/213465768-SSL-Certificate-Guide#common-keytool-commands>

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 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 \$data-dir/etc/ssl/keystore.jks which contains the Jetty SSL certificate to use. Instructions are available on the [Eclipse Jetty documentation](#)⁷¹³ site or our [SSL Certificate Guide](#)⁷¹⁴.
2. Edit \$data-dir/etc/nexus.properties and save the file after making the following changes.
 - a. Add a new line containing:
`application-port-ssl=8443`

Change 8443 to 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.

- b. Uncomment the line containing nexus-args by removing the leading # and space characters. Change the comma delimited nexus-args property value to include \${jetty.etc}/jetty-https.xml. 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`
- c. (Optional) If you do not wish to expose a plain http connector, you can remove \${jetty.etc}/jetty-http.xml, part of the nexus-args value.
- d. Add a new line containing:

⁷¹³ <https://www.eclipse.org/jetty/documentation/9.4.x/configuring-ssl.html>

⁷¹⁴ <https://support.sonatype.com/hc/en-us/articles/213465768-SSL-Certificate-Guide>

```
ssl.etc=${karaf.data}/etc/ssl
```

3. Edit \$install-dir/etc/jetty/jetty-https.xml
 - a. Set your keystore passwords appropriately. Make sure all three passwords are the same value (required). If you use "password" as your keystore and *PrivateKeyEntry* password, this step is not necessary.
 - b. (Recommended) Before the line containing <Set name="KeyStorePath"> add a new line containing the alias name of your keystore file *PrivateKeyEntry*. For example if your *PrivateKeyEntry* is named "jetty", add a line like this:

```
<Set name="certAlias">jetty</Set>
```
4. Restart the repository manager. Verify HTTPS connections can be established. If NXRM starts successfully, but you receive an error connecting to the secure port, then refer to our [Troubleshooting Eclipse Jetty SSL Certificates](#)⁷¹⁵ article.
5. Update the Base URL to use https in your repository manager configuration using the [Base URL\(see page 263\)](#) capability.

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.

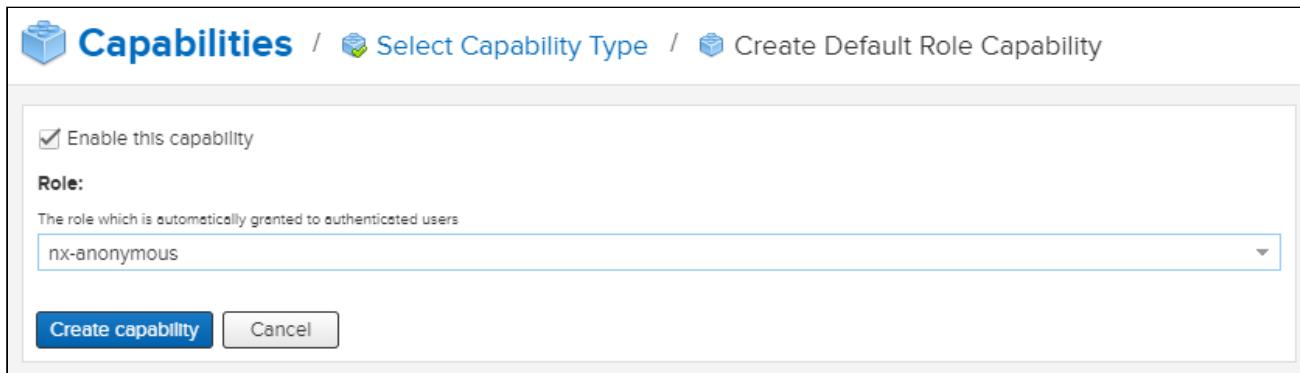
Default Role

 Available in Nexus Repository OSS and Nexus Repository Pro

To enable appending a default role to all authenticated users, create a [new Capability](#)⁷¹⁶ using capability type *Default Role* as pictured below, you will then be able to select the role that you want applied to users.

⁷¹⁵ <https://support.sonatype.com/hc/en-us/articles/225023368-Troubleshooting-Eclipse-Jetty-SSL-Certificates>

⁷¹⁶ <https://help.sonatype.com/display/NXRM3M/System+Configuration#SystemConfiguration-AccessingandConfiguringCapabilities>



Once this is saved, the *Default Role Realm* will be added to the active list of security realms, and start applying the new role to all authenticated users.

- ⓘ This default role is appended to authenticated users dynamically, and will *not* show up as assigned to any user via the *User settings* page.

Atlassian Crowd Support

- ⓘ Available in Nexus Repository Manager Pro only

Atlassian Crowd is a single sign-on and identity management product that many organizations use to consolidate user accounts and control which users and groups have access to which applications. Atlassian Crowd support is a feature preinstalled and ready to configure in Nexus Repository Manager Pro. Nexus Repository Manager contains a security realm that allows you to configure the repository manager to authenticate against an Atlassian Crowd instance.

- ❗ Using LDAP and Crowd Realms together in the repository manager may work, but this is not supported. If you already use LDAP support, we recommend adding your LDAP server as a Crowd directory accessible to the Crowd application instead of using both LDAP and Crowd realms in the repository manager.

Topics in this section:

- [Preparing Atlassian Crowd\(see page 348\)](#)
- [Configure Crowd Integration\(see page 349\)](#)

Preparing Atlassian Crowd

Compatibility

Always use the latest version of Crowd available at the time your version of Nexus Repository Manager Pro was installed or upgraded. If upgrading to a newer Crowd server, carefully review the Crowd server release notes for REST API backwards compatibility issues.

Crowd support in Nexus Repository Manager Pro only works in Crowd versions that support the Crowd REST API. Older versions use a deprecated SOAP-based API and are less reliable and performant.

Crowd support is actively tested with the highest available version of Crowd at the time Nexus Repository Manager Pro is released.

Adding a New Application to the Crowd Server

- ⓘ These instructions are a general guide to adding an application to Crowd. For current detailed instructions, visit the [official Crowd documentation](#)⁷¹⁷.

To connect Nexus Repository Manager to Crowd, you will need to configure Nexus Repository Manager as an application in Crowd.

1. Login to Crowd as a user with administrative rights.
2. Click on the Applications tab.
3. Click Add application to display a form.

Next, create the new application with the following values in the Details section:

- *Application Type*
- *Name*
- *Description*

Description is optional. Choose a password for the application. The repository manager will use this password to authenticate with the Crowd server. Confirm the password, then click the *Next* button to fill out *Connection*.

On the *Connection* screen provide the URL and the remote IP address for your repository manager application. You can click *Resolve IP Address*, which prompts Crowd to resolve the IP address for your application.

⁷¹⁷ <https://confluence.atlassian.com/display/CROWD/Adding+an+Application>

Once you have completed the *Connection* form, click on *Next* to advance to the *Directories* form.

The *Directories* form allows you to select the user directory used for authentication.

Click *Next* to advance to the *Authorisation* form. If any of the directories selected in the previous form contain groups, each group is displayed in a dropdown menu. You can check the *Allow all users to authenticate* box above the dropdown within the directory, or you can select specific groups that are allowed to authenticate to Nexus Repository Manager Pro via Crowd. This option would be used if you wanted to limit repository manager access to specific sub-groups within a larger Crowd directory. If your entire organization is stored in a single Crowd directory, you may want to limit repository manager access to a group that contains only developers and administrators.

Click *Next* to advance to the final screen, *Confirmation*, which gives you a summary of your Crowd server settings. Click *Add application* to confirm the settings.

Configure Crowd Integration

Enable the Crowd Capability

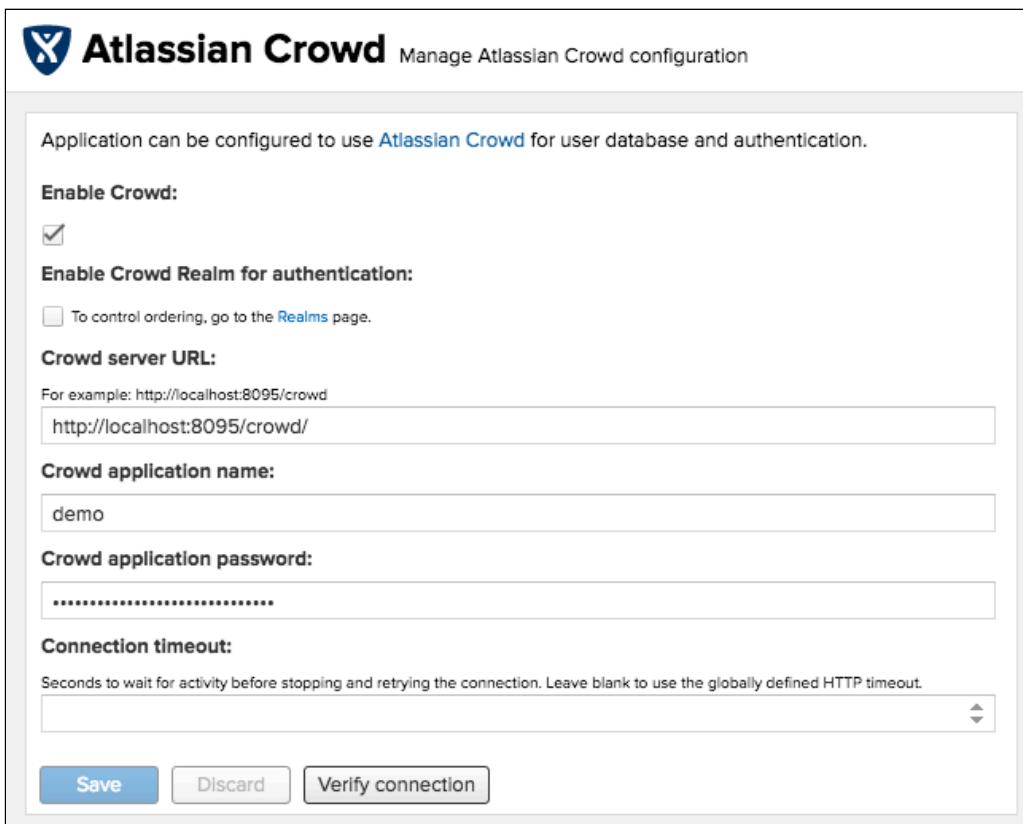
To enable Crowd perform the following steps:

1. Select *Capabilities* to open the *Capabilities* panel, located in the *Administration* menu under *System*
2. Click the *Create capability* button to get to the *Select Capability Type* table
3. Select *Crowd* to open the *Create Crowd Capability* panel
4. Complete the form by entering the *Crowd Server URL* and the *Application Name* and *Application Password* that correspond to your Crowd application

This form also includes an option to *Use the Nexus truststore*. You would check this box if you configured and want to manage Crowd with the HTTPS protocol, mentioned in [Configure Pro to Trust Crowd's Secure URL](#)(see page 350).

After you create the capability, you will see the *Enable Crowd* box checked automatically in the *Atlassian Crowd* panel in the *Administration* menu under *Security*. Further, you can see the *Crowd server URL*, *Crowd application name* and *Crowd application password*, all automatically filled in. Additionally here, you can configure *Connection timeout*, a value that specifies the number of milliseconds the repository manager will wait for a response from Crowd. A value of zero indicates that there is no timeout limit. Leave the field blank to use the default timeout.

You can use the *Verify Connection* button to confirm your connection to Crowd is working. Pressing *Save* will save any changes made to the Crowd configuration.



The screenshot shows the 'Manage Atlassian Crowd configuration' page. It includes fields for enabling Crowd, Crowd Realm authentication, Crowd server URL (set to http://localhost:8095/crowd), Crowd application name (set to demo), Crowd application password (redacted), and a connection timeout dropdown set to 10 seconds. Buttons at the bottom include Save, Discard, and Verify connection.

Configure Pro to Trust Crowd's Secure URL (Optional)

Although optional, we advise the connection from Nexus Repository Manager Pro to your Crowd server to use the HTTPS protocol.

If the Crowd certificate is not signed by a public certificate authority, you may have to explicitly trust the server certificate as explained in [Outbound SSL - Trusting SSL Certificates of Remote Repositories\(see page 0\)](#). A common symptom observed is the peer not authenticated message, when trying to connect to the untrusted Crowd server.

Adding the Crowd Server Certificate to the Truststore

In order to add the server certificate of your Crowd server to the truststore, go to *SSL Certificates*, located under *Security* in the *Administration* menu. In the *SSL Certificates* panel click the *Load Certificate* button, which prompts a dropdown menu with two options:

- *Load from server*: where you can enter the full https:// URL from the Crowd server
- *Paste PEM*: where you can enter an encoded, remote certificate generated from Crowd

Read more about centralizing [SSL certificates\(see page 340\)](#) to the Nexus Repository Manager in [Security\(see page 309\)](#).

Configure Nexus Repository Manager Pro Crowd Security

There are two approaches available to manage what privileges a Crowd user has when they log in to the repository manager. You can map Crowd groups to roles or map Crowd users to roles.

- ⓘ Mapping Crowd groups to Nexus Repository Manager Pro roles is preferred because there is less configuration involved overall in Nexus Repository Manager Pro and assigning users to Crowd groups can be centrally managed inside of Crowd by your security team after the initial repository manager setup.

Mapping a Crowd Group to Roles

When mapping a Crowd group to a Nexus Repository Manager Pro role, you are specifying the permissions (via roles) that users within the Crowd group will have after they authenticate.

To map a Crowd group to a Nexus Repository Manager Pro role, open the *Roles* panel by clicking on the *Roles* link under *Security* in the *Administration* panel. Click on *Create role* button, select *External Role Mapping*, then click *Crowd*. This will take you *Create Role* panel, as mentioned in [Roles\(see page 317\)](#).

After choosing the Crowd realm, the Role drop-down should list all the Crowd groups to which the Crowd application has access. Select the group you would like to map in the *Role* field.

- ⓘ If you have two or more groups in a Crowd application with identical names but in different directories, the repository manager will only list the first one that Crowd finds. Therefore, Crowd administrators should avoid identically named groups in Crowd directories.

Before you save, you must add at least one role or privilege to the mapped group. After you have them added using the *>* button or drag and drop to the *Contained* or *Given* areas (respectively), click the *Save* button.

Saved mappings will appear in the list of roles with a mapping value of *Crowd*.

Mapping a Crowd User to Roles

Consider the Crowd server user with an id of `johnsmith`. In the Crowd administrative interface, the `johnsmith` Crowd realm user as a member of both `dev` and `crowd-administrators` groups.

To add an external user go to the *Administration* menu in the repository manager, then click *Users* in the *Security* section.

Click the *Source* dropdown button and select *Crowd*. To search for users from the Crowd realm you can either enter an individual username within the filter box, or click the magnifying glass icon to generate the list of all users from the Crowd realm.

When the name you entered appears, click on the row of the name you desire to create the mapping for. This will take you to a form where you can assign available roles. You must map at least one role to the Crowd managed user in order to Save.

Staging

 Available in Nexus Repository Manager Pro only

Overview

In modern software development, it is imperative to thoroughly test software before it is deployed to a production system or externally accessible repository. Mostly commonly a release candidate will first be deployed to a staging system which is a close copy of the production system so it can undergo a series of rigorous tests before a decision is made to promote it to production or return it to development.

The staging functionality in Nexus Repository Manager Pro supports promotion of software components matching your organization's software development life cycle phases by moving those components between repositories. This allows the creation of isolated release candidates that can be discarded or promoted to make it possible to support the decisions that go into certifying a release.

Building Blocks

The basic building blocks of the staging functionality consist of hosted repositories, component tags, and the ability to move and delete components between those hosted repositories directly via a REST API. The following briefly describes each of these building blocks:

- Hosted repositories - Components/assets are uploaded to hosted repositories and can then be promoted/moved to other hosted repositories or deleted.
- Component tags - Uploaded components/assets are tagged so that they can be identified as a logical group and be transferred together. See [Tagging\(see page 357\)](#) for more information.
- REST endpoints
 - Move - allows all components and assets matching a specified search query to be moved to a target repository.
 - Delete - allows for the deletion of all components and assets matching a specified search query.

- ❗ There are some [usage limitations](#)(see page 356) to the current staging functionality which should be reviewed prior to use in your workflows.

Workflow

As depicted in the figure below, *Typical Staging Workflow*, a typical staging workflow will involve some hosted repositories (along with possibly some group repositories to include other dependencies) and an external system which is coordinating the promotion process. This usually includes a CI system, such as Jenkins, which will continuously build each commit, as well as a release manager who will promote some builds for testing and ultimately release.

Build Phase

When a developer pushes new code, the CI system will pick it up and perform the build automatically including any unit or integration tests. The resulting artifacts are then uploaded to the `incoming` NXRM repository. As part of this operation the artifacts will be tagged so they can later be identified. Commonly the CI project/job ID and the build ID can be used as the tag. For example, the job `project-abc` with the build ID 142 would result in the tag `project-abc-build-142`.

Test Phase

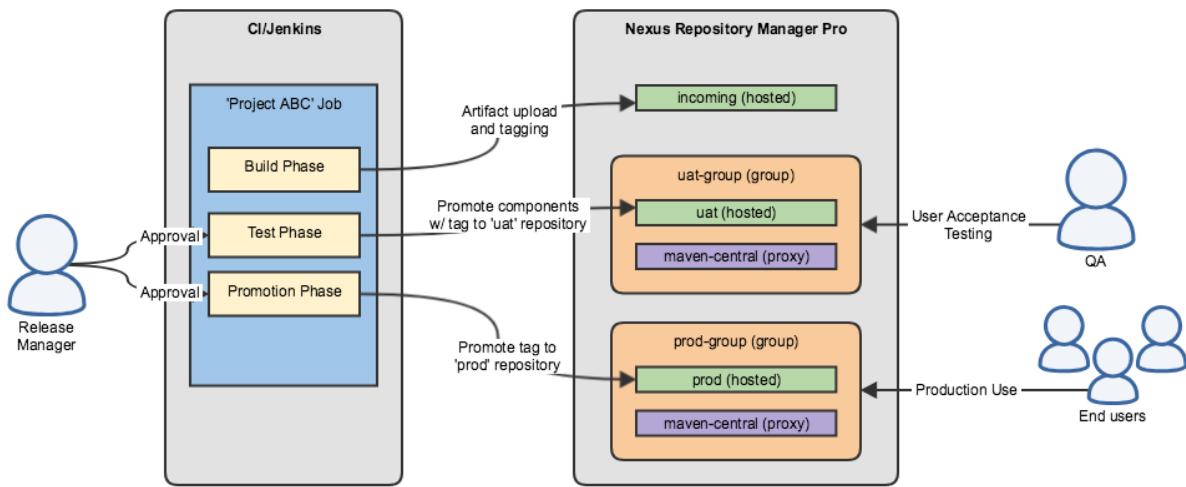
When the release manager decides, any given build can be promoted to the next stage. As shown in the figure below, a build would be promoted from the `incoming` hosted repository to the `uat` repository in preparation for testing. This could be initiated via a button in the CI job/pipeline or perhaps a separate job. NXRM will receive the promotion call to take all components tagged with `project-abc-build-142` and move them into the `uat` repository. This repository is available to the testing group of users so that they can ensure quality.

Release Phase

When testing is complete, the build can then be promoted to the next stage which in our example is production. As shown in the figure below, a build would be promoted from the `uat` repository to the `prod` repository in preparation for release. This again may be initiated via button in the CI job/pipeline or a separate job. NXRM will receive the promotion call to take all components tagged with `project-abc-build-142` and move them into the `prod` repository.

If the build fails testing, then the release manager can instead issue the call to delete all components tagged with `project-abc-build-142`.

Typical Staging Workflow



REST Endpoints

See the [REST and Integration API](#)(see page 464) page for more details on using these endpoints via the interactive UI.

Promotion

```
POST service/rest/v1/staging/move/{repository}
```

This endpoint allows us to promote (or move) components that match a search into the specified repository.

```
curl -u admin:admin123 -X POST 'http://127.0.0.1:8081/service/rest/v1/staging/move/uat?tag=project-abc-142'
```

This example will move all components tagged with project-abc-142 into the repository uat. Note that any search criteria can be used though within the context of staging the tag parameter is most common. See [Search API](#)(see page 489) for more.

The response will include a payload that shows the components that were moved.

Example Response

```
{  
  "status": 200,  
  "message": "Move Successful",  
  "data": {  
    "destination": "uat",  
    "components moved": [  
      {  
        "name": "project-abc",  
        "group": "com.mycompany",  
        "version": "2.1.1"  
      }  
    ]  
  }  
}
```

Deletion

```
POST /service/rest/v1/staging/delete
```

This endpoint allows us to delete components that match a search from a repository. The primary use case within the context of staging is if a set of components has been promoted to a test environment, and then failed testing, it can be deleted from the system.

```
curl -u admin:admin123 -X POST 'http://127.0.0.1:8081/service/rest/v1/staging/delete?tag=project-abc-142'
```

This will remove all the components with the tag project-abc-142. As with Promote, any search criteria can be used though within the context of staging the tag parameter is most common. See [Search API](#)⁷¹⁸ for more.

The response will include a payload that shows the components that were deleted.

⁷¹⁸ <https://help.sonatype.com/display/NXRM3M/Search+API>

Example Response

```
{  
  "status": 200,  
  "message": "Delete Successful",  
  "data": {  
    "components deleted": [  
      {  
        "repository": "uat",  
        "group": "com.mycompany",  
        "name": "project-abc",  
        "version": "2.1.1"  
      }  
    ]  
  }  
}
```

Integrations

The Sonatype [Nexus Platform Plugin for Jenkins](#)⁷¹⁹ includes staging related build steps to help you implement staging workflows in your builds.

NXRM 2 Migration

Migration from NXRM 2 staging to NXRM 3 staging is not supported. The staging feature set in NXRM 3 has been completely redesigned and is not compatible with NXRM 2 staging. The rationale behind this redesign was to provide a set of flexible building block capabilities that can be combined and orchestrated as needed to accommodate your organization's software build and release pipeline. The set of REST endpoints that expose these capabilities allow for easy integration into CI/CD systems while not being prescriptive of the workflows or client tooling and technologies needed to interact with them.

[See our guide](#)⁷²⁰ for more details about NXRM 2 and 3 staging differences.

Limitations

- Only hosted repositories are supported.
- Only APT, Docker, Maven, npm, NuGet, Raw and Yum formats are currently supported.
- You can only transfer between repositories of the same policy. You cannot transfer from a snapshot repository to a release repository.

⁷¹⁹ <https://help.sonatype.com/display/NXI/Nexus+Platform+Plugin+for+Jenkins>

⁷²⁰ <https://guides.sonatype.com/repo3/technical-guides/staging-v2-v3-upgrade/>

Tagging

 Available in Nexus Repository Manager Pro Only

Overview

Tagging is a feature available in Nexus Repository Manager Pro that provides the ability to mark a set of components with a tag so they can be logically associated to each other. The usage of the tags is up to you but the most common scenarios would be a CI build ID for a project (e.g. project-abc-build-142) or a higher level release train when you are coordinating multiple projects together as a single unit (e.g. release-train-13). Tagging is used extensively by the [Staging\(see page 352\)](#) feature.

It should be noted that tags are applied at the component level and not to individual assets.

Endpoints

Tagging capabilities have been made available by way of a REST API. For more details on how to access and use these endpoints via an interactive UI please see the [REST and Integration API\(see page 464\)](#) page.

Add Tag

```
POST /service/rest/v1/tags
```

This endpoint allows us to create a tag. Here is a simple example:

```
curl -u admin:admin123 -X POST --header 'Content-Type: application/json' http://127.0.0.1:8081/service/rest/v1/tags \
-d '{
  "name": "project-abc-142",
  "attributes": {
    "jvm": "9",
    "built-by": "jenkins"
  }
}'
```

The name field cannot exceed 256 characters and can only contain letters, numbers, underscores, hyphens and dots and cannot start with an underscore or dot. The attributes field can contain any valid JSON data that might assist you. There is a maximum size of 20k for the attributes field.

The response contains two additional timestamp fields to represent when the tag was first created and last updated. These two fields are set automatically and are ignored if sent in a POST or PUT.

Example Response

```
{  
  "name": "project-abc-142",  
  "attributes": {  
    "jvm": "9",  
    "built-by": "jenkins"  
  },  
  "firstCreated": "2017-06-12T22:42:55.019+0000",  
  "lastUpdated": "2017-06-12T22:42:55.019+0000"  
}
```

List Tags

```
GET /service/rest/v1/tags
```

This endpoint allows us to iterate through a listing of all the tags.

This endpoint has no parameters:

```
curl -u admin:admin123 -X GET http://127.0.0.1:8081/service/rest/v1/tags
```

Typically this will produce a response that is the first page of many tags, as such:

Example Response

```
{  
  "items": [  
    {  
      "name": "project-abc-142",  
      "attributes": {  
        "jvm": "9",  
        "built-by": "jenkins"  
      },  
      "firstCreated": "2017-06-12T22:42:55.019+0000",  
      "lastUpdated": "2017-06-12T22:42:55.019+0000"  
    },  
    {  
      "name": "project-abc-456",  
      "attributes": {  
        "jvm": "9",  
        "built-by": "jenkins"  
      },  
      "firstCreated": "2017-06-13T22:24:55.019+0000",  
      "lastUpdated": "2017-06-13T22:24:55.019+0000"  
    }  
  ],  
  "continuationToken": null  
}
```

This endpoint uses a [pagination](#)⁷²¹ strategy that can be used to iterate through all the tags if desired.

Note that the ordering of the tags is consistent across multiple queries but it is not alphabetical.

UI Tip

It is also possible to see components associated with a given tag via the UI. You can do this by navigating to the *Tags* section of the *Browse* menu, selecting a tag and clicking *Find Tagged Components*. This same behavior is doable by doing a custom search using the *Tag* criteria.

Get Tag

```
GET /service/rest/v1/tags/{tagName}
```

This endpoint allows us to get details about an individual tag.

```
curl -u admin:admin123 -X GET http://127.0.0.1:8081/service/rest/v1/tags/project-abc-142
```

⁷²¹ <https://help.sonatype.com/display/NXRM3M/Pagination>

Example Response

```
{  
  "name": "project-abc-142",  
  "attributes": {  
    "jvm": "9",  
    "built-by": "jenkins"  
  },  
  "firstCreated": "2017-06-12T22:42:55.019+0000",  
  "lastUpdated": "2017-06-12T22:42:55.019+0000"  
}
```

Update Tag

```
PUT /service/rest/v1/tags/{tagName}
```

This endpoint allows us to update the attributes of a single tag. Note that the tag name cannot be changed, nor can the timestamp fields.

```
curl -u admin:admin123 -X PUT --header 'Content-Type: application/json' http://127.0.0.1:8081/service/rest/v1/tags/project-abc-142 -d '{  
  "attributes": {  
    "jvm": "9",  
    "built-by": "bob"  
  }  
'
```

Example Response

```
{  
  "name": "project-abc-142",  
  "attributes": {  
    "jvm": "9",  
    "built-by": "bob"  
  },  
  "firstCreated": "2017-06-12T22:42:55.019+0000",  
  "lastUpdated": "2017-06-13T21:42:55.019+0000"  
}
```

Associate Components with a Tag

```
POST /service/rest/v1/tags/associate/{tagName}
```

This endpoint allows you to search for components and associate them to an existing tag. Note that this endpoint does not create a tag if it does not exist.

```
curl -u admin:admin123 -X POST 'http://127.0.0.1:8081/service/rest/v1/tags/associate/project-abc-142?repository=my-company&group=com.mycompany&name=project-abc&version=2.1.1'
```

In this example we are searching within a repository for a specific artifact using the group, name, and version. See [Search API](#)(see page 489) for details on how to search for components.

Example Response

```
{  
  "status": 200,  
  "message": "Association successful",  
  "data": {  
    "components associated": [  
      {  
        "name": "project-abc",  
        "group": "com.mycompany",  
        "version": "2.1.0"  
      }  
    ]  
  }  
}
```

The response contains a collection of all the components that were successfully associated to the tag. This response can be used to assert that the association worked as expected.

Tagging During Component Upload

Tagging components can also be accomplished by including a tag when exercising the upload component REST endpoint provided by the [Components API](#).⁷²² In the example below, we have specified the tag to be associated with the component being uploaded by adding "-F tag=project-abc-142" to the component upload call. Additionally, components can also be uploaded and tagged by way of the [user interface](#)⁷²³.

⁷²² <https://help.sonatype.com/display/NXRM3M/Components+API>

⁷²³ <https://help.sonatype.com/display/NXRM3M/Uploading+Components>

```
curl -v -u admin:admin123 -X POST 'http://localhost:8081/service/rest/v1/components?repository=my-company' \
-F groupId=com.mycompany \
-F artifactId=project-abc \
-F version=2.1.1 \
-F asset1=@project-abc-2.1.1.jar \
-F asset1.extension=jar \
-F tag=project-abc-142
```

Disassociate Components from a Tag

```
DELETE /service/rest/v1/tags/associate/{tagName}
```

This endpoint allows you to search for components and disassociate them from a tag. Note that this endpoint is the same as the association endpoint, it just uses a DELETE instead of a POST.

```
curl -u admin:admin123 -X DELETE 'http://127.0.0.1:8081/service/rest/v1/tags/associate/project-abc-142?repository=my-company&group=com.mycompany&name=project-abc&version=2.1.1'
```

Example Response

```
{
  "status": 200,
  "message": "Disassociation successful",
  "data": {
    "components_disassociated": [
      {
        "name": "project-abc",
        "group": "com.mycompany",
        "version": "2.1.1"
      }
    ]
  }
}
```

Privileges

Tagging is controlled by the following privileges.

nx-tags-all

Allows all tagging actions.

nx-tags-associate

Allows the association of tags to components located in repositories the user has browse privileges to as well as the viewing of tags.

nx-tags-create

Allows the creation and viewing of tags.

nx-tags-delete

Allows the deletion and viewing of tags.

nx-tags-disassociate

Allows the disassociation of tags from components located in repositories the user has browse privileges to as well as the viewing of tags.

nx-tags-read

Allows the viewing of tags.

nx-tags-update

Allows the update and viewing of tags.

Cleanup Task

Having a CI system automatically tag every build can result in an overabundance of tags. The *Admin - Cleanup tags* task allows admins to delete tags based on the following criteria:

- Age (in days) of when the tag was first created (default is 0 days)
- Age (in days) of when the tag was last modified (default is 0 days)
- A regular expression on the tag name (e.g. project-abc-*)

If multiple options are selected then both criteria will need to match on the tag. Additionally you can select to delete the components associated with the tag. This option can be restricted to a specific repository or format.

 In situations where a component has two tags associated with it and the tag clean up task has been configured to delete the components, that component will be deleted even when the clean up task only matches on one of the tags. For example, if component 'A' is tagged with 'build-123' and 'dev-build-123' and the tag clean up task was configured to clean up tag 'dev-build-123' AND delete components associated with it, component 'A' will be deleted. Tag 'build-123' being associated with the component does not prevent the deletion.

The task log for the *Admin - Cleanup tags* task contains the tags (and components if selected) that have been deleted. See the [Task Logging\(see page 279\)](#) section on the System Configuration page for more details.

Best Practices

The following are some best practice recommendations for using the tagging functionality:

- Use short tag names and attributes to maximize performance of tag related operations and storage utilization. Maintaining a large number of tags with attributes of maximum allowable size (20k) could result in undesirable performance degradation.
- Proactively use the *Admin - Cleanup tags* task to clean up unused tags. The *Admin - Cleanup tags* task provides configuration options to support routine removal of aged or unused tags and the associated components (if desired).
- Use a tag naming scheme that will ensure uniqueness of created tags and prevent naming collisions and potential interruptions to CI pipeline workflows.

Formats

Nexus Repository Manager supports an evolving list of official and community repository formats. Formats are free to use, regardless of product license.

 Sonatype Official Formats		 Sonatype Community Formats	
Official formats are included in all Nexus Repository Manager distributions(see page 22) and actively maintained and supported under a Pro license⁷²⁴ .		Community formats are open source projects developed by the Sonatype Nexus Community⁷²⁵ . Use of these formats require manual installation and updates from an open source project.	
Format	Description	In-product or Community	HA-C
APK⁷²⁶	Proxy Alpine OS⁷²⁷ apk ⁷²⁸ packages and cache them in repository manager.		no

⁷²⁴ <https://www.sonatype.com/nexus-repository-sonatype>

⁷²⁵ <https://github.com/sonatype-nexus-community>

⁷²⁶ <https://github.com/sonatype-nexus-community/nexus-repository-apk>

⁷²⁷ <https://alpinelinux.org/>

⁷²⁸ https://wiki.alpinelinux.org/wiki/Alpine_Linux_package_management

APT (see page 367)	Use Advanced Package Tool (APT) tools such as apt-get to access hosted Debian, Ubuntu and other Linux software packages.		no
Bower (see page 371)	Keep track of all your web site frontend development packages using Bower registries.		yes
Cargo ⁷²⁹	Package repositories for the Rust ⁷³⁰ package manager Cargo ⁷³¹ .		no
Chef ⁷³²	Provision Chef Cookbooks ⁷³³ using tools like Knife ⁷³⁴ and Berkshelf ⁷³⁵ .		no
CocoaPod S (see page 376)	Dependency manager for Swift and Objective-C Cocoa projects.		no
Composer ⁷³⁶	PHP Composer ⁷³⁷ dependency management for the PHP programming language.		no
Conan (see page 378)	Share your C/C++ packages in a central repository.		no
Conda (see page 381)	Proxy Conda packages for languages such as Python, R, Ruby, Lua, Scala, Java, JavaScript, C/C++ and FORTRAN.		no
CPAN ⁷³⁸	Perl programmers share modules using the Comprehensive Perl Archive Network (CPAN). Consolidate module access securely through repository manager.		no
Docker (see page 383)	Proxy popular Docker registries such as DockerHub and host your own private images securely.		yes
ELPA ⁷³⁹	Emacs users unite! Proxy an Emacs Lisp Package Archive repository.		no
Git LFS (see page 403)	Store large files such as audio samples, videos, datasets, and graphics inside repositories and use simple text pointers to these inside your Git project.		yes

⁷²⁹ <https://github.com/sonatype-nexus-community/nexus-repository-cargo>

⁷³⁰ <https://www.rust-lang.org/>

⁷³¹ <https://doc.rust-lang.org/cargo/index.html>

⁷³² <https://github.com/sonatype-nexus-community/nexus-repository-chef>

⁷³³ <https://docs.chef.io/cookbooks.html>

⁷³⁴ <https://downloads.chef.io/chef-dk/>

⁷³⁵ <http://berkshelf.com/>

⁷³⁶ <https://github.com/sonatype-nexus-community/nexus-repository-composer>

⁷³⁷ <https://getcomposer.org>

⁷³⁸ <https://github.com/sonatype-nexus-community/nexus-repository-cpan>

⁷³⁹ <https://github.com/sonatype-nexus-community/nexus-repository-elpa>

Go(see page 406)	Modernize your development process and reuse your Go ⁷⁴⁰ (golang) modules by sharing them in repositories.		no
Helm(see page 409)	Manage packages for Kubernetes by accessing Helm Charts in Helm ⁷⁴¹ repositories.		no
Maven(see page 411)	Leverage the most experienced Maven repository format product to host your private Java components and proxy defacto public repositories like Central using tooling such as Maven, Ant and Gradle.		yes
npm(see page 422)	Publish your javascript node.js projects to hosted registries and integrate dependencies from external JavaScript package registries.		yes
NuGet(see page 428)	Use NuGet client compatible tooling to push and install .Net packages. Development is easier using consolidated hosted and proxy registries.		yes
p2⁷⁴²	Proxy p2 format repositories for your Eclipse IDE and other Equinox based application dependencies.		no
Puppet⁷⁴³	Securely serve your Puppet Modules in repository manager by proxying repositories like Puppet Forge ⁷⁴⁴ .		no
PyPI(see page 440)	Python development tools such as pip and twine can be used install and publish packages to PyPI repositories.		yes
R(see page 445)	Proxy packages from the official R registry ⁷⁴⁵ and host your own packages.		no
Raw(see page 448)	A flexible format that does not enforce any type of layout enabling you to store and serve any type of binary content such as Maven sites over HTTP.		yes
Rubygems (see page 454)	Host your own gems and proxy remote gem repositories like rubygems.org ⁷⁴⁶ .		yes
Yum(see page 460)	First class support for hosting and proxying your RPM distributions.		yes

⁷⁴⁰ <https://golang.org>⁷⁴¹ <https://helm.sh/>⁷⁴² <https://help.sonatype.com/display/NXRM3M/p2+Repositories>⁷⁴³ <https://github.com/sonatype-nexus-community/nexus-repository-puppet>⁷⁴⁴ <https://forge.puppet.com/>⁷⁴⁵ <https://cran.r-project.org/>⁷⁴⁶ <https://www.rubygems.org/>

Apt Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

Debian and systems based on it like Ubuntu, Linux Mint, and Raspbian use the same package management system. APT⁷⁴⁷ (Advanced Package Tool) is a set of tools for managing Debian packages, and therefore the applications installed on your Debian system. It provides a wide set of operations like searching repositories, installing packages with their dependencies, and managing upgrades.

Nexus Repository Manager supports the Apt repository format for proxy repositories as well as for hosted repositories. Proxy repositories give you several significant advantages. You are no longer dependent on the availability of public repositories and an external network connection. When packages are cached, you can snapshot metadata and be sure all packages will be available even when removed from the remote repository. With hosted repositories, you have reliable storage for your own and third-party packages.

 This format is disabled by default in NXRM Pro HA-C environments.

Proxying Apt Repositories

You can set up a Apt proxy repository to access a remote repository location.

To proxy a Apt repository, you simply create a new *apt (proxy)* as documented in [Repository Management](#)⁷⁴⁸.

Minimal configuration steps are:

- Define *Name* e.g. apt-proxy
- Define URL for *Remote storage*, e.g. `http://archive.ubuntu.com/ubuntu/`
- Define the *Distribution* e.g. bionic
- If the remote repository has a flat format, check the *Flat* checkbox
- Pick a *Blob store* for *Storage*

⁷⁴⁷ <https://wiki.debian.org/Apt>

⁷⁴⁸ <https://help.sonatype.com/repomanager3/configuration/repository-management>

⚠ We do not create a default Apt proxy repository as there are many. Determine which repositories are appropriate for your environment.

Hosting Apt Repositories

A hosted repository for Apt can be used to upload your packages as well as third-party packages. To host Apt packages, create a new *apt (hosted)* repository as documented in [Repository Management](#)⁷⁴⁹.

When creating a Apt Hosted repository you need to generate a PGP signing key pair or use an existing one. The current implementation of Apt hosted repository signs only metadata. Nexus Repository Manager does not sign packages. If you want to use signed .deb and .udeb packages, upload to the repository already signed files.

Generate a key pair in a Linux system with the following commands:

```
apt-get update
apt-get install gpg
gpg --gen-key
gpg --list-keys
cd <path to the folder to import the key pair>
gpg --armor --output public.gpg.key --export <gpg key Id>
gpg --armor --output private.gpg.key --export-secret-key <gpg key Id>
```

A key ID looks like: '515F58C16D58E682E91ACEFF17B5C97F9A816AD7'

Minimal configuration steps are:

- Define *Name* e.g. apt-hosted
- Define the *Distribution* e.g. bionic
- Put the private pgp key into *Signing Key* field, as described above
- Put the passphrase for the private signing key into the *Passphrase* field if you have defined one
- Pick a *Blob store for Storage*

To use an Apt hosted repository you need to export the gpg public key into your Linux system. Use the public key (`public.gpg.key`) from the key pair generated above.

```
apt-get update
apt-get install gnupg
apt-key add <full folder path in the container>/public.gpg.key
```

⁷⁴⁹ <https://help.sonatype.com/repomanager3/configuration/repository-management>

Deploying Packages to Apt Hosted Repositories

You can use HTTP POST or use *Upload* in the UI, as documented [here](#)(see page 230), to upload packages to Apt hosted repository.

The following example uses the `curl` command and example credentials of `admin` for user and `admin123` for password to upload a `test.deb` file to a hosted Apt repository:

```
curl -u "admin:admin123" -H "Content-Type: multipart/form-data" --data-binary "@./test.deb" "http://localhost:8081/repository/apt-hosted/"
```

Browsing Apt Repositories and Searching Packages

You can browse Apt repositories in the user interface inspecting the components and assets and their details, as described in [Browsing Repositories and Repository Groups](#)⁷⁵⁰.

Searching for Apt packages can be performed in the user interface, too. It finds all packages that are currently stored in the repository manager, as described in [Searching for Components](#)⁷⁵¹.

Configuring Apt Client

If you already loaded a metadata using `apt update` commands first clean it by removing all files from `/var/lib/apt/lists/`.

To configure the Apt client to work with Nexus Repository Manager edit the file `/etc/apt/sources.list`. Add the following line if you want to add the repository to the list, or replace the content of the file if you're going to use only your repository:

```
deb <repository URL> <distribution> main
```

For a hosted repository you should use the `<distribution>` from the repository properties. For a proxy repository, the `<distribution>` should be the same as in the original remote repository settings.

You can get the `<repository URL>` from the table in [Browsing Repositories and Repository Groups](#)(see page 227) via the UI.

⁷⁵⁰ <https://help.sonatype.com/repomanager3/user-interface/browsing-repositories-and-repository-groups>

⁷⁵¹ <https://help.sonatype.com/repomanager3/user-interface/searching-for-components>

Taking a Snapshot of a repository metadata

A snapshot in terms of the Apt repository is a named static copy of metadata of the repository. Changes in the repository will not cause any changes in the snapshot. A snapshot does NOT contain any binary files. So be ready if the content of the repository changed the snapshot can include some invalid metadata.

Available snapshots can be viewed in [UI Browse\(see page 227\)](#) in the `/snapshots` folder. Files of a particular snapshot are available in `/snapshots/<snapshot Id>` folder.

Snapshots functionality is available for both proxy and hosted repositories.

You can use a snapshot to set up an apt client. The URL of the snapshot is `<URL of the repository>/snapshots/<snapshot Id>`. For example:

```
http://localhost:8081/repository/apt-proxy/snapshots/release123.
```

Snapshots functionality is useful in following cases:

- Create a snapshot and use it as a repository for a proxy repository. In this case, you are sure that packages will not be updated or removed because of changes in the remote repository.

 To use the removed in the remote repository package, this package must be cached in the proxy repository. If not, you are aware about changed remote repository, the updated package will not be installed.

- Use snapshots for the hosted repository to freeze versions of all packages. If someone has changed the content of the repository, you are aware about changed remote repository, the updated package will not be installed.

There are two variants to create a snapshot:

- To snapshot all metadata of the repository use a HTTP MKCOL request. The following example uses the curl command and example credentials of `admin` for user and `admin123` for password to create a snapshot with the id `release123`:

```
curl -u "admin:admin123" -X MKCOL "http://localhost:8081/repository/apt-proxy/snapshots/release123"
```

- To snapshot filtered by [architectures](#)⁷⁵² and [components](#)⁷⁵³ metadata use HTTP PUT requests with the appropriate data, as follows:

1. Create a filter. This is a text file with the following format:

⁷⁵² <https://wiki.debian.org/DebianRepository/Format#Architectures>

⁷⁵³ <https://wiki.debian.org/DebianRepository/Format#Components>

```
Architectures: <list of architectures>
Components: <list of components>
```

For example:

```
Architectures: i386
Components: restricted universe
```

2. Use HTTP PUT request to snapshot a filtered metadata. This example uses `curl` to create a snapshot of repository `apt-proxy` with filter stored in `filter.txt` and id `filtered_snapshot`:

```
curl -u "admin:admin123" -X PUT -H "Content-Type: multipart/form-data" -T "filter.txt" "http://localhost:8081/repository/apt-proxy/snapshots/filtered_snapshot"
```

Bower Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

Bower⁷⁵⁴ is a package manager for front-end web development. JavaScript developers using Bower gain convenient access to a large amount of packages from the remote Bower registry. This reduces the complexity of their development efforts and improves the resulting applications.

Nexus Repository Manager Pro and Nexus Repository Manager OSS support the Bower registry format for hosted and proxy repositories. This allows the repository manager to take advantage of the packages in the official Bower registry and other public registries without incurring repeated downloads of packages.

The official Bower registry is available for searches at <http://bower.io/search> and for package retrieval via the URL

<https://registry.bower.io/>.

You can publish your own packages to a private Bower registry as a hosted repository on the repository manager and then expose the remote and private repositories to Bower as a repository group, which is a repository that merges and exposes the contents of multiple repositories in one convenient URL. This allows you to reduce time and bandwidth usage for accessing Bower packages a registry as well as share your packages within your organization in a hosted repository.

⁷⁵⁴ <http://bower.io/>

Proxying Bower Repositories

You can set up a Bower proxy repository to access a remote repository location, for example the official Bower registry at

`https://registry.bower.io`
that is configured as the default on Bower.

To proxy a Bower registry, you simply create a new *bower (proxy)* as documented in [Repository Management](#)(see page 234) in detail.

Minimal configuration steps are:

- Define *Name*
- Define URL for *Remote storage* e.g.
`https://registry.bower.io`
- Select a *Blob store* for *Storage*

The Bower specific configuration section include the setting to Enable rewrite of package URLs. This causes Bower to retrieve components and their dependencies through the repository manager even if original metadata has hard-coded URLs to remote repositories. This setting Force Bower to retrieve packages via the proxy repository is enabled by default.

If deactivated, no rewrite of the URL occurs. As a result, the original component URL is exposed. Turning off rewrite capabilities proxies the information directly from the remote registry without redirecting to the repository manager to retrieve components.

Hosting Bower Repositories

Creating a Bower hosted repository allows you to register packages in the repository manager. The hosted repository acts as an authoritative location for these components. This effectively creates an asset that becomes a pointer to an external URL (such as a Git repository).

To add a hosted Bower repository, create a new repository with the recipe *bower (hosted)* as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name* e.g. `bower-internal`
- Select *Blob store* for *Storage*

Bower Repository Groups

A repository group is the recommended way to expose all your Bower repositories from the repository manager to your users, with minimal additional client side configuration. A repository group allows you to

expose the aggregated content of multiple proxy and hosted repositories as well as other repository groups with one URL in tool configuration. This is possible for Bower repositories by creating a new repository with the *bower (group)* recipe as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define Name e.g. bower-all
- Select *Blob store* for Storage
- Add Bower repositories to the *Members* list in the desired order

Installing Bower

Bower is typically installed with npm. Since the repository manager supports NPM repositories for proxying, we recommend to configure the relevant NPM repositories and npm as documented in [npm Registry](#)(see page 422) prior to installing Bower. Once this is completed you can install Bower with the usual command.

```
npm install -g bower
```

Bower version 1.5 or higher is required and can be verified with

```
$ bower -v  
1.7.7
```

In addition Bower requires a custom URL resolver to allow integration with Nexus Repository Manager Pro and Nexus Repository Manager OSS. The resolver is an API introduced in Bower version 1.5. Bower fetches component and version information through the repository manager, then automatically searches and saves the component in the repository. You can install the resolver with:

```
npm install -g bower-nexus3-resolver
```

Alternatively you can install the resolver on a per-project basis instead by adding it as a dependency in your package.json:

```
"devDependencies" : {  
  "bower-nexus3-resolver" : "*"  
}
```

Configuring Bower Package Download

Once you have set up your repositories for Bower packages, and installed Bower and the custom resolver, you can create a .bowerrc JSON file to access registry URLs. The registry value is configured to access

the Bower repository group that exposes the proxy and hosted repositories together. The resolvers configuration is necessary, so that Bower uses the required custom resolver.

Global `.bowerrc` file in your home directory for package download via the group `bower-all`

```
{  
  "registry" : {  
    "search" : [ "http://localhost:8081/repository/bower-all" ]  
  },  
  "resolvers" : [ "bower-nexus3-resolver" ]  
}
```

- ⓘ The `.bowerrc` file can be located in various locations. For global configuration for a specific developer working on multiple projects the users home directory is a suitable location. If multiple files exist, they are merged. Details can be found [in Bower configuration documentation](#)⁷⁵⁵.

With this configuration in place, any further Bower command invocations trigger package downloads via the repository manager.

Running an `install` command logs the download via the repository manager:

```
$ bower install jquery  
bower jquery#*  
  not-cached nexus+http://localhost:8081/repository/bower-all/jquery#*  
bower jquery#*  
  resolve nexus+http://localhost:8081/repository/bower-all/jquery#*  

```

If anonymous access to the repository manager is disabled, you have to specify the credentials for the accessing the repository manager as part of the URL like `http://username:password@host:port/repository/bower-all` and add a `nexus` section to your `.bowerrc` file.

```
{  
  "nexus" : {  
    "username" : "myusername",  
    "password" : "mypassword"  
  }  
}
```

⁷⁵⁵ <https://bower.io/docs/config/>

Downloaded packages are cached, do not have to be retrieved from the remote repositories again and can be inspected in the user interface.

Browsing Bower Repositories and Searching Packages

You can browse Bower repositories in the user interface inspecting the components and assets and their details, as described in [Searching for Components](#)(see page 213).

Searching for Bower packages can be performed in the user interface, too. It finds all packages that are currently stored in the repository manager, either because they have been pushed to a hosted repository or they have been proxied from an upstream repository and cached in the repository manager.

Registering Bower Packages

If you are authoring your own packages and want to distribute them to other users in your organization, you have to register them to a hosted repository on the repository manager. This establishes a metadata file in the repository that links to the source code repository. Typically this is a git repository. The consumers can then download it via the repository group as documented in [Configuring Bower Package Download](#)(see page 373).

You can specify the URL for the target hosted repository in the `register` value in your `.bowerrc` file. If you are registering all packages you create in the same hosted repository you can configure in the your global configuration file e.g. located in your users home directory:

```
{  
  "registry" : {  
    "search" : [  
      "http://localhost:8081/repository/bower-all"  
    ],  
    "register" : "http://localhost:8081/repository/bower-internal"  
  },  
  "resolvers" : [ "bower-nexus3-resolver" ]  
}
```

Alternatively, if you desire to use a per-project `.bowerrc` file that you potentially version in your source code management system with the rest of the package code, you can use a simplified file:

```
"registry": {  
  "register": "http://localhost:8081/repository/bower-internal"  
}
```

Authentication is managed in the same manner as for proxying with anonymous access disabled as documented in [Configuring Bower Package Download](#)(see page 373), e.g. `"register": "http://`

admin:admin123@localhost:8081/repository/bower-hosted". With this configuration you can run a command such as:

```
bower register example-package git://gitserver/project.git
```

All semantic version tags on the git repository are now exposed as version for this package and consumers can install the package via the repository group like any other package.

```
bower install example-package
```

CocoaPods Repositories

Introduction

CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects.

 This format is disabled by default in NXRM Pro HA-C environments.

 Because of a limitation in the spec for Cocoapods, NXRM does not currently check authentication for spec files. This means that an informed user could potentially **list** the contents of proxy repositories without authenticating. This will be resolved in a future release. For more information see <https://github.com/CocoaPods/CocoaPods/issues/9151>.

Support information

- CocoaPods client support: 1.7.2 and greater
- Cocoapods client OS: MacOS
- Spec repositories: CDN only. Content delivery network mirror of master Specs repository <https://github.com/cocoapods/specs> is <https://cdn.cocoapods.org/>. It is possible to work with custom Specs repositories only after CDN mirror creation.
- Pods repositories: GitHub, Bitbucket, direct http(s) link to archive with source code

- SSL: CocoaPods client can only work with CDN mirror by https protocol. Http protocol is not supported. See [configuration details](#)⁷⁵⁶ for more about setting up SSL in NXRM.
- Authentication to remote resources: authentication to remote cdn spec repositories and pod repositories not supported.

Proxying CocoaPods Repositories

NXRM CocoaPods proxy repository caches metadata .podspec.json files from CDN mirror of Specs repository. Link to such mirrors should be configured in the *Remote storage* field. After fetching of Spec file, NXRM parses the source link from the metadata file. Such sources can be GitHub, BitBucket or direct https link. Pod library source code will be cached as an archive file such as tar.gz or zip. Https source will be downloaded directly while in the case of GitHub or BitBucket source, the Pod library will be downloaded by REST API call.

To proxy a CocoaPods repository, you simply create a new *CocoaPods (proxy)* as documented in [Repository Management](#)⁷⁵⁷.

Minimal configuration steps are:

- Define *Name* e.g. cocoapods-proxy
- Define URL for *Remote storage* e.g. <https://cdn.cocoapods.org/>
- Pick a *Blob store for Storage*

Configure CocoaPods client

In Swift or Objective-C Cocoa project source folder, update Podfile and set the source as a link to NXRM CocoaPods repository. Example:

```
$cat Podfile
require 'cocoapods'
# Uncomment the next line to define a global platform for your project
# platform :ios, '9.0'
source 'https://localhost:8443/repository/cocoapods-proxy/'
target 'CocoaPodsTest' do
use_frameworks!

pod 'Alamofire', '~>5.0.0-beta.5'
pod 'Igor'
pod 'SafetySDK'
pod 'DSTestDR'
end
```

⁷⁵⁶<https://help.sonatype.com/repomanager3/security/configuring-ssl#ConfiguringSSL-InboundSSL-ConfiguringtoServeContentviaHTTPS>

⁷⁵⁷ <https://help.sonatype.com/repomanager3/configuration/repository-management>

Configure SSL

CocoaPods client can work with CDN mirror only by https protocol. One of two points below should be configured:

- Configure NXRM to use trusted SSL certificate. Self signed ones will not work.
- As the CocoaPods client uses the `curl` command to download Pod files from NXRM, setup curl to work with self signed certificates by adding the `--insecure` option into the `.curlrc` file at your home directory. If no such file exists, then just create it. Example:

```
$cat ~/.curlrc  
--insecure
```

Configure NXRM Authentication

The CocoaPods client uses the `curl` command to download Pod files from NXRM. Therefore if anonymous access is not enabled on the NXRM side, then you need to configure authentication for curl.

Configure `loginandpassword` in `.netrc` file in the home folder of your workstation and use `themachine` option to configure hostname or IP address. If you haven't such file then just create it. Example:

```
$cat ~/.netrc  
machine localhost  
login admin  
password admin123
```

Conan Repositories

NEW IN RELEASE 3.22

Introduction

Conan is decentralized, portable, extensible and table package manager for C/C++ projects with client-server architecture. Conan allows to encapsulate C/C++ project dependencies, distribute them, and consume them in other projects. This involves the complex challenges of transitive dependencies, versioning, licensing, and so forth. It works across Linux, OSX and Windows platforms.

ⓘ This format is disabled by default in NXRM Pro HA-C environments.

Supported features

- Repository types - Proxy

Conan client setup

The

pip

tool is recommended way to install the Conan client:

```
pip install conan
```

To discover different approaches or any additional details of Conan client installation use Conan documentation: <https://docs.conan.io/en/latest/installation.html>

Proxying Conan Repositories

You can set up a Conan proxy repository to access a remote repository location, for example

<https://conan.bintray.com>

(the default one).

To proxy a Conan repository, you simply create a new `conan (proxy)` as documented in [Repository Management](#)⁷⁵⁸ in detail.

Minimal configuration steps are:

- Define *Name*
- Define URL for *Remote storage* e.g.

<https://conan.bintray.com>

- Select a *Blob store for Storage*

⁷⁵⁸ <https://help.sonatype.com/display/NXRM3M/Repository+Management>



- If you are going use HTTPs protocol for Conan and your NXRM is configured to use a self-signed certificate then it is needed to add your certificate into trusted list. It can be done by adding certificate into <user_home>/.conan/cacert.pem file. It is simple txt file that can be changed by any text editor.
- If you disabled Anonymous access, then it is needed to enable the *Conan Bearer Token Realm* (via *Administration* → *Security* → *Realms*): add Conan Bearer Token Realm. Details on activating realms can be found [here](#)(see page 310).

Conan client configuration to work with NXRM

To avoid bypassing Conan NXRM repository and stop Conan package installation directly from remote servers, you need to delete the default remotes . To delete any remote server use the command:

```
conan remote remove <remote name>
```

where <remote name> is the remote's name.

For example to delete the default one:

```
conan remote remove conan-center
```

To add new remote with link to NXRM Conan repository use command:

```
conan remote add <remote name> <remote URL> <SSL flag>
```

- <remote name> is remote name, can be equal to NXRM Conan repository name
- <remote URL> is NXRM Conan repository URL
- <SSL flag> is true to enable HTTPs protocol or false to enable HTTP protocol

For example example to add Conan proxy repository with conan-proxy name:

```
conan remote add conan-proxy http://localhost:8081/repository/conan-proxy/ false
```

To change remote configuration (change remote URL or enable/disable SSL) use the command:

```
conan remote update <remote name> <remote URL> <SSL flag>
```

You can review list of configured remotes by following the command:

```
conan remote list
```

Conda Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

NEW IN RELEASE 3.19

Introduction

"Package, dependency and environment management for any language--Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN

Conda⁷⁵⁹ is an open-source package management system and environment management system that runs on Windows, macOS, and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads, and switches between environments on your local computer. It was created for Python programs but it can package and distribute software for any language."

You can create a proxy repository in Nexus Repository Manager that will cache packages from a remote anaconda repository, like [Continuum](#)⁷⁶⁰, [Conda-Forge](#)⁷⁶¹ and [Anaconda](#)⁷⁶². Then, you can make the Conda client use your Nexus Repository proxy instead of the remote repository.

 This format is disabled by default in NXRM Pro HA-C environments.

Supported features

- Conda versions - 4.6 and newer
- Conda package formats -
 - .conda
 - and
 - tar.bz2

⁷⁵⁹ <https://docs.conda.io/projects/conda/en/latest/index.html>

⁷⁶⁰ <https://repo.continuum.io/pkgs>

⁷⁶¹ <https://conda.anaconda.org/conda-forge>

⁷⁶² <https://conda.anaconda.org/anaconda/>

- Repository types - Proxy

Repository configuration

Proxying the Conda repository

To proxy a Conda repository, you simply create a new *conda (proxy)* as documented in [Repository Management](#)⁷⁶³ in detail.

Minimal configuration steps are:

- Define *Name* - e.g. conda-proxy
- Define *URL for Remote storage* - e.g.
<https://repo.continuum.io/pkgs/>
- Select a *Blob store for Storage*

Configuring the Conda client to use NXRM

Using the Conda client, you can now download packages from your NXRM Conda proxy like so:

```
$ conda install -c <user>:<password>@<hostname>:<port>/repository/<repository_name> numpy
```

The command above tells Conda to fetch (and install) packages from your NXRM Conda proxy. The NXRM Conda proxy will download any missing packages from the remote Conda repository and cache the packages on the NXRM Conda proxy. The next time any client requests the same package from your NXRM Conda proxy, the already cached package will be returned to the client.

If you are using the Conda client only with NXRM, you can add channels to the Conda repositories in the `.condarc` file:

```
$ conda config --add channels <user>:<password>@<hostname>:<port>/repository/<repository_name>
```

After these changes, you can install packages without directly specifying a channel, like this:

```
$ conda install numpy
```

⁷⁶³ <https://help.sonatype.com/repomanager3/configuration/repository-management>

Docker Registry

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

Docker containers and their usage have revolutionized the way applications and the underlying operating system are packaged and deployed to development, testing and production systems.

The creation of the [Open Container Initiative](#)⁷⁶⁴, and the involvement of a large number of stakeholders, guarantees that the ecosystem of tools around the lightweight containers and their usage will continue to flourish.

[Docker Hub](#)⁷⁶⁵ is the original registry for Docker container images and it is being joined by more and more other publicly available registries such as the [Google Container Registry](#)⁷⁶⁶ and others.

Nexus Repository Manager Pro and Nexus Repository Manager OSS support Docker registries as the Docker repository format for hosted and proxy repositories. You can expose these repositories to the client-side tools directly or as a repository group, which is a repository that merges and exposes the contents of multiple repositories in one convenient URL. This allows you to reduce time and bandwidth usage for accessing Docker images in a registry as well as share your images within your organization in a hosted repository. Users can then launch containers based on those images, resulting in a completely private Docker registry with all the features available in the repository manager.

-  The minimum version of Docker that works with Nexus Repository is version 1.8. But please note that Docker is a fast moving project and requires usage of current operating system versions and tools. For example, usage of Red Hat Enterprise Linux 6 is simply not supported. Please use the [official documentation](#)⁷⁶⁷ as reference and help for your usage.

Topics in this section:

- [SSL and Repository Connector Configuration](#)(see page 384)
- [Proxy Repository for Docker](#)(see page 386)
- [Hosted Repository for Docker \(Private Registry for Docker\)](#)(see page 387)
- [Grouping Docker Repositories](#)(see page 388)
- [Content Selectors and Docker](#)(see page 388)

⁷⁶⁴ <https://www.opencontainers.org/>

⁷⁶⁵ <https://hub.docker.com/>

⁷⁶⁶ <https://cloud.google.com/container-registry/>

⁷⁶⁷ <https://docs.docker.com/>

- [Authentication\(see page 394\)](#)
- [Accessing Repositories\(see page 395\)](#)
- [Searching\(see page 396\)](#)
- [Pulling Images\(see page 397\)](#)
- [Pushing Images\(see page 398\)](#)
- [Scaling Repositories\(see page 399\)](#)
- [Foreign Layers\(see page 400\)](#)
- [Docker Content Trust\(see page 402\)](#)

SSL and Repository Connector Configuration

Docker relies on secure connections using SSL to connect to the repositories. You are therefore required to expose the repository manager to your client tools via HTTPS. This can be configured via an external proxy server, which can also be used to [scale your repositories\(see page 399\)](#), or directly with the repository manager. Further details can be found in [Inbound SSL - Configuring to Serve Content via HTTPS\(see page 344\)](#).

Interaction of the docker client with repositories requires specific ports to be used. These can be configured in the repository configuration in the Repository Connectors section. In order for this to work on your network, you need to ensure that the chosen ports are available in your organization and not used by some other application, and that no firewall or other network configuration prevents connectivity.

 The docker client does not allow a context as part of the path to a registry, as the namespace and image name are embedded in the URLs it uses. This is why requests to repositories on the repository manager are served on a specific and separate port from the rest of the application instead of how most other repositories serve content via a path i.e. <nexus-hostname>/<repositoryName>/<path to content> .

The recommended minimal configuration requires one port for a Docker repository group used for read access to all repositories and one port for each hosted Docker repository that will receive push events from your users. The Repository Connectors configuration, displayed in *Figure: “Repository Connector Configuration”*, is available in the configuration for proxy and hosted Docker repositories as well as Docker repository groups.

Repository Connectors

Connectors allow Docker clients to connect directly to hosted registries, but are not always required. Consult our [documentation](#) for which connector is appropriate for your use case.

HTTP:

Create an HTTP connector at specified port. Normally used if the server is behind a secure proxy.



HTTPS:

Create an HTTPS connector at specified port. Normally used if the server is configured for https.



18079



Figure: Repository Connector Configuration

If you have configured the repository manager to use HTTPS directly, you have to configure a HTTPS repository connector. If an external proxy server translates incoming HTTPS requests to HTTP and forwards the request to the repository manager via HTTP you have to configure the respective HTTP port.

- ⓘ A configured context-path for the user interface does not affect the repository connector URLs used by Docker. E.g. if your repository manager instance is configured to be available at `http://localhost:8081/nexus` instead of the default root context `http://localhost:8081/`, the URLs for your Docker repositories will still only use the configured port for the repository and omit the context path in the URL. This is a side-effect of the fact that Docker does not support context paths in the registry API.

Tips for SSL Certificate Usage

Nexus Repository Manager is not configured with HTTPS connectors by default as it requires an SSL certificate to be generated and configured manually.

The requirement of Docker to use HTTPS forces the usage of SSL certificates. By default, Docker looks up the validity of the certificate by checking with certificate authorities. If you purchased a certificate that is registered with these authorities, all functionality works as desired.

If you create a certificate yourself with tools such as `openssl`, it is self-signed and not registered. Using a self-signed certificate requires further configuration steps to ensure that Docker can explicitly trust it.

- ⚠ Docker Daemon can stand up instances with the `--insecure-registry` flag to skip validation of a self-signed certificate. But the repository manager does not support the use of the flag, as it generates known bugs and other implementation issues.

To generate a trustworthy self-signed certificate for the repository manager use `keytool`, a utility that lets you manage your own private key pairs and certificates. See our [knowledge base article](#)⁷⁶⁸ to learn how to configure the utility.

Support for Docker Registry API

The Docker client tools interact with a repository via the registry API. It is available in version 1 (V1) and version 2 (V2). The newer V2 will completely replace the old V1 in the future. Currently Docker Hub and other registries as well as other tools use V2, but in many cases fall back to V1. E.g., search is currently only implemented in V1.

Nexus Repository Manager supports V1 as well as V2 of the API. All Docker repository configurations contain a section to configure Docker Registry API Support . If you activate Enable Docker V1 API for a repository it is enabled to use V1 as a fallback from V2. Without this option any V1 requests result in errors from the client tool.

- ⓘ Generally V1 support is only needed for repository groups that will be used for command line-based searches, when any client side tools in use require V1 or when a upstream proxy repository requires V1. If you are unsure if your setup uses these or V1, it is recommended to activate V1 support as there should be no harm if it is not needed.

Proxy Repository for Docker

Docker Hub is the common registry used by all image creators and consumers. To reduce duplicate downloads and improve download speeds for your developers and CI servers, you should proxy Docker Hub and any other registry you use for Docker images.

To proxy a Docker registry, you simply create a new `docker (proxy)` as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define Name
- Define URL for Remote storage
- Enable Docker V1 API support, if required by the remote repository

⁷⁶⁸ <https://support.sonatype.com/hc/en-us/articles/217542177>

- Select correct *Docker index*, further configure *Location of Docker index* if needed
- Select *Blob store* for Storage

If you intend users to access the proxy repository directly, you may wish to configure a Repository Connector as explained in [SSL and Repository Connector Configuration](#)(see page 384). This is not necessary for proxy content that is accessed via a Docker group.

The *Remote Storage* has to be set to the URL of the remote registry you want to proxy. The configuration for proxying Docker Hub uses the URL <https://registry-1.docker.io> for the Remote storage URL.

The Proxy configuration for a Docker proxy repository includes a configuration URL to access the *Docker Index*. The index is used for requests related to searches, users, docker tokens and other aspects. The registry the index are typically co-hosted by the same provider, but can use different URLs. E.g. the index for Docker Hub is exposed at <https://index.docker.io/>. The default option of *Use proxy registry (specified above)* will attempt to retrieve any index data from the same URL configured as the Remote storage URL. The option to *Use Docker Hub* fulfills any index related requests by querying the Docker Hub index at <https://index.docker.io/>. This configuration is desired when the proxy repository is Docker Hub itself or any of its mirrors. The option to use a *Custom index* allows you to specify the URL of the index for the remote repository. It is important to configure a correct pair of Remote Storage URL and Docker Index URL. In case of a mismatch, search results potentially do not reflect the content of the remote repository and other problems can occur.

Just to recap, in order to configure a proxy for Docker Hub you configure the Remote Storage URL to <https://registry-1.docker.io>, enable Docker V1 API support and for the choice of Docker Index select the *Use Docker Hub* option.

Hosted Repository for Docker (Private Registry for Docker)

A hosted repository using the Docker repository format is typically called a private Docker registry. It can be used to upload your own container images as well as third-party images. It is common practice to create two separate hosted repositories for these purposes.

To create a Docker hosted repository, simply create a new *docker (hosted)* repository as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Select *Blob store* for Storage

If you add a Repository Connectors configuration as documented in [SSL and Repository Connector Configuration](#)(see page 384) you can push images to this repository, and subsequently access them directly from the hosted repository or ideally from the Docker repository group as documented in [Grouping Docker Repositories](#)(see page 388).

By default this setup will allow repeated deployment of images. If you want to enforce new deployments using different versions, set the *Deployment Policy* to *Disable Redeploy*.

An additional configuration option for Disable Redeploy to *Allow redeploy only on 'latest' tag* enables only the latest tag to be redeployed while continuing to disable all others.

Grouping Docker Repositories

A repository group is the recommended way to expose all your repositories for read access to your users. It allows you to pull images from all repositories in the group without needing any further client side configuration after the initial setup. A repository group allows you to expose the aggregated content of multiple proxy and hosted repositories with one URL to your tools.

To create a Docker repository group, simply create a new *docker (group)* repository as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define Name
- Select *Blob store* for Storage
- Add Docker repositories to the *Members* list in the desired order

Typically the member list includes a mixture of proxy and hosted repositories to allow access to public as well as private images.

Using the Repository Connectors port of the repository group and the URL of the repository manager in your client tool gives you access to the container images in all repositories from the group. Any new images added as well as any new repositories added to the group will automatically be available.

 Check out this repository configuration demonstrated in [this video](#)⁷⁶⁹.

Content Selectors and Docker

Administrators of the Docker format may desire to not share all Docker components across all teams. In general, [Content Selectors](#)(see page 251) provide you with a way to select specific content from a whole and, in conjunction with [Security](#)⁷⁷⁰ (specifically the privileges created by Content Selectors), are the recommended way to manage this desire. However, the Docker format has some non-standard complexity to setting up security against the CLI and UI which is communicated here.

A basic assumption this page makes is that you are able to sort the security of your docker images by name. Thus below we use the word `team` in our examples where `team` is the representation of that which you are sorting. For example, it might be `sonatype/nexus` for this product/project team, or further `sonatype /`

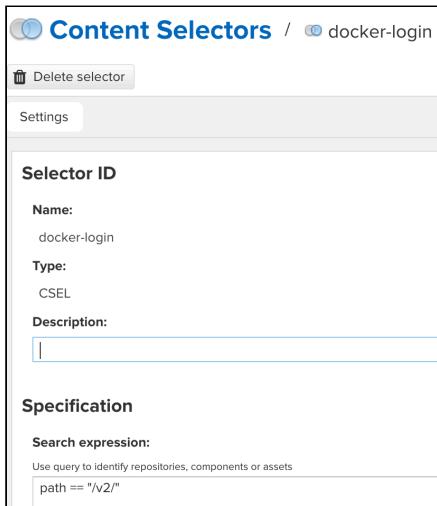
⁷⁶⁹ <https://www.youtube.com/watch?v=oxCztw5MfAw>

⁷⁷⁰ <https://help.sonatype.com/display/NXRM3M/Security?src=contextnavpagetreemode>

nexus/docker for the docker specific parts. If you are unable to do this, this page likely will not help you or your solution will be much more complex than the scenarios given.

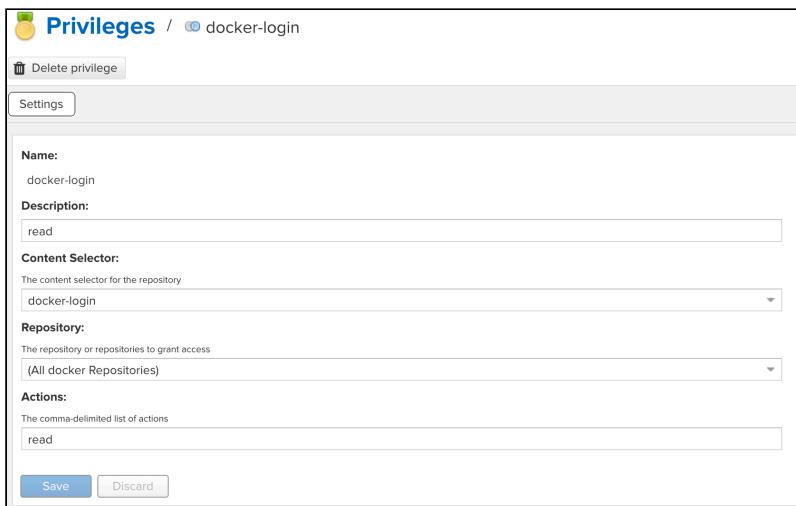
docker login

This command requires access to the /v2/ path and can be established by creating a content selector with the search expression path == "/v2/" as shown here:



The screenshot shows the 'Content Selectors' configuration page. A new selector named 'docker-login' is being created. The 'Specification' section contains a search expression 'path == "/v2/"'. This selector is intended to identify the Docker repository's /v2/ endpoint.

Once the content selector is created, a *Repository Content Selector* privilege with Actions defined with read should be created, as shown below, then assigned to a role and subsequently users.

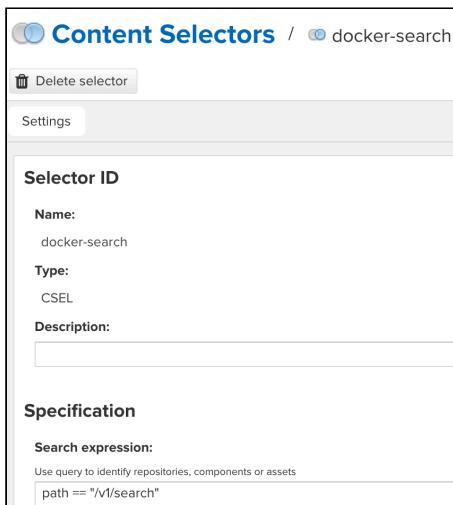


The screenshot shows the 'Privileges' configuration page. A new privilege named 'docker-login' is being created with the 'read' action. It is associated with the previously created 'docker-login' content selector and applies to all Docker repositories.

Access to login likely will need to be combined with other privileges to be useful.

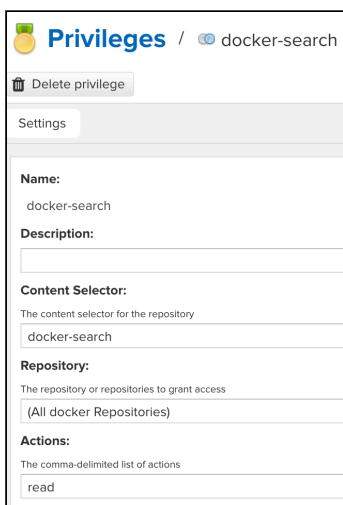
docker search

This command requires access to the /v1/search path and can be established by creating a content selector with the search expression path == "/v1/search" as shown here:



The screenshot shows the 'Content Selectors' page in the Nexus Repository Manager. A single content selector named 'docker-search' is listed. The 'Specification' section contains the search expression 'path == "/v1/search"'.

Once the content selector is created, a *Repository Content Selector* privilege with *Actions* defined as `read` should be created, as shown below, then assigned to a role and subsequently users.



The screenshot shows the 'Privileges' page in the Nexus Repository Manager. A single privilege named 'docker-search' is listed. It includes a 'Content Selector' field set to 'docker-search', a 'Repository' field set to '(All docker Repositories)', and an 'Actions' field set to 'read'.

Disclaimers

This privilege gives those assigned access to search the entire repository which can be more than you have access to pull/push.

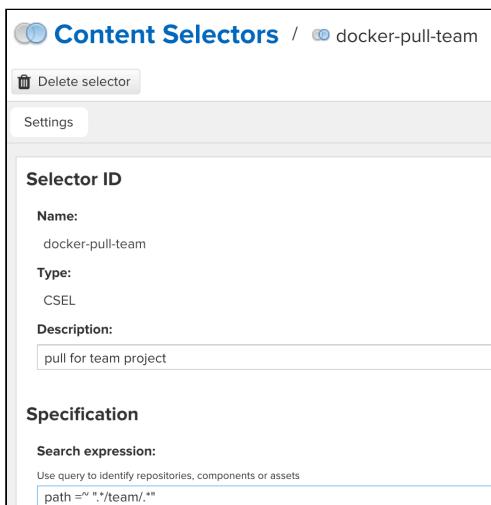
Further, this privilege does not give you access to search anything via UI, instead it is delegating search via the CLI. How to give permission to search via the UI is defined [below](#)(see page 393).

docker pull (by team)

This command requires that the content selector path be restricted by team. This can be done loosely (e.g. path \sim ".*/team/.*) with "contains team" or strictly (e.g. path $=^$ "/v2/team/") with "starts with team". Note that for simpler docker packages, you can restrict using library too (e.g. path $=^$ "/v2/library/hello-world/").

Any of these methods give you the ability to pull any version of components from team.

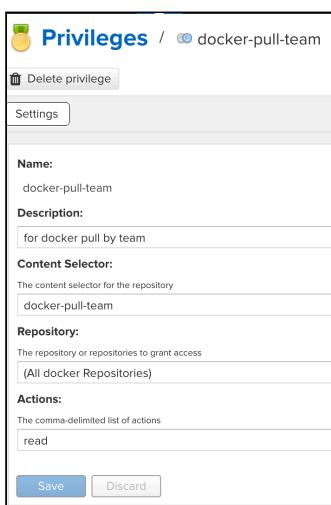
Here's an example content selector:



The screenshot shows the 'Content Selectors' page with a single entry:

- Name:** docker-pull-team
- Type:** CSEL
- Description:** pull for team project
- Specification:** path $=^$ ".*/team/.*"

Once the content selector is created, a *Repository Content Selector* privilege with *Actions* defined as `read` should be created, as shown below, then assigned to a role and subsequently users.



The screenshot shows the 'Privileges' page with a single entry:

- Name:** docker-pull-team
- Description:** for docker pull by team
- Content Selector:** docker-pull-team
- Repository:** (All docker Repositories)
- Actions:** read

Content selectors and privileges will need created for each "team".

- i** With security setup in this way, users trying docker pull commands by hash will get unauthorized responses because the hash does not fall in the pattern of the path defined above.

i **Tip: Browse UI and Content Selector Preview**

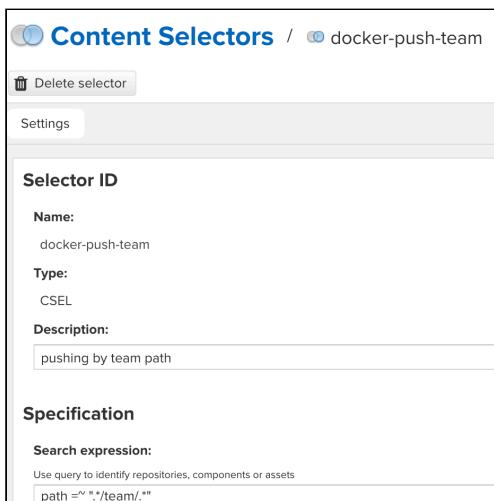
read action grants permission to the Browse UI but only to manifests and tags (which match the path defined). Individual layers will not be shown.

The same will reflect in Content Selector preview results.

docker push (by team)

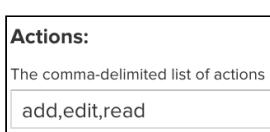
Like *docker pull*, this command also requires that the content selector path be restricted by team. This can be done loosely (e.g. path =~ ".*/team/.") with "contains team" or strictly (e.g. path =^ "/v2/team/") with "starts with team". Using the simpler library method from *docker pull* (e.g. path =^ "/v2/library/hello-world/") will not work.

Because the exact same paths can be used if you have *docker pull* content selectors too, you can reuse them if you've implemented *docker pull*. However you need them distinct, you'd create a new content selector like this:



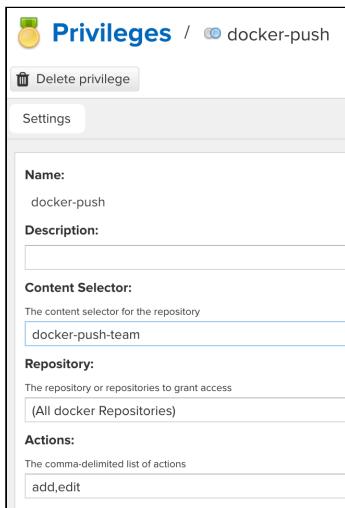
The screenshot shows the 'Content Selectors' configuration page. A new content selector named 'docker-push-team' has been created. The 'Selector ID' section includes fields for Name (set to 'docker-push-team'), Type (set to 'CSEL'), and Description (set to 'pushing by team path'). The 'Specification' section contains a search expression: 'path =~ ".*/team/.*"'.

If you are combining permissions, edit your existing privilege *Actions* to put (comma delimited) add and edit (in addition to read) like so:



The screenshot shows the 'Actions' configuration page. The 'Actions:' field contains 'add,edit,read'. The 'The comma-delimited list of actions' field also contains 'add,edit,read'.

If you are not combining, create a new a *Repository Content Selector* privilege with *Actions* defined as `add,edit` like so:



The screenshot shows the 'Privileges' creation interface. The title bar says 'Privileges / docker-push'. The form fields are as follows:

- Delete privilege** (button)
- Settings** (button)
- Name:** docker-push
- Description:** (empty text area)
- Content Selector:** The content selector for the repository
docker-push-team
- Repository:** The repository or repositories to grant access
(All docker Repositories)
- Actions:** The comma-delimited list of actions
add,edit

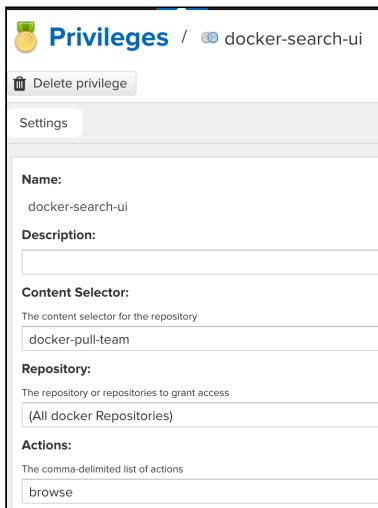
Then assign the privilege to roles and subsequently users.

Content selectors and privileges will need created (or combined/edited) for each "team".

Search UI

The permissions described in [Docker Search\(see page 390\)](#) do not grant permission to the *Search UI*. To gain permission to the *Search UI* the role needs to have the *nx-search-read* privilege assigned. In addition to this to see the items restricted by path, the Action of *browse* needs to be added. You can do this by adding (comma delineated) to an existing privilege with the path for your team (e.g. `path =^ "/v2/team/`, `path =~ ".*/team/.*`" or `path =^ "/v2/library/hello-world/`) or by creating a new privilege then assigning that to roles.

Here's an example of the new privilege you would create:



The screenshot shows the 'Privileges' creation interface for 'docker-search-ui'. The form fields are as follows:

- Delete privilege** (button)
- Settings** (button)
- Name:** docker-search-ui
- Description:** (empty text area)
- Content Selector:** The content selector for the repository
docker-pull-team
- Repository:** The repository or repositories to grant access
(All docker Repositories)
- Actions:** The comma-delimited list of actions
browse

If creating a new privilege, you would assign this privilege to roles and then the roles to users as needed. If editing the existing privilege, permissions will be granted when you save.

Authentication

Configuration Guidance

In some cases in order to docker login and in order to access docker anonymously, you also need to enable the *Docker Bearer Token Realm* as generally outlined in [Realms\(see page 310\)](#). This realm is inactive by default.

If access to a repository requires the user to be authenticated, docker will check for authentication access in the .docker/config.json file. If authentication is not found, some actions will prompt for authentication but otherwise a docker login command will be required before the actions can be performed. Typically this is required when anonymous access to the repository manager is disabled or the operation requires authentication.

The docker login command observes the following syntax for the desired repository or repository group:

```
docker login <nexus-hostname>:<repository-port>
```

Provide your repository manager credentials of username and password as well as an email address. This authentication is persisted in ~/.docker/config.json and reused for any subsequent interactions against that repository. Individual login operations must be performed for each repository and repository group you want to access in an authenticated manner.

Specifically when planning to push to a repository a preemptive login operation is advisable as it removes the need for user interaction and is therefore suitable for continuous integration server setups and automated scenarios.

Anonymous Read Access

By default when using Nexus Repository Manager, all docker repositories require authentication to be read from using the command line tools regardless of any permissions granted by the [Anonymous\(see page 323\)](#) user (if enabled) or, in the case of proxy repositories, the remotes' settings. For Docker in NXRM, this can be bypassed on a per repository basis by editing the repository settings and enabling the *Allow anonymous docker pull* checkbox under the *Repository Connectors* section shown at the bottom of *Figure: "Repository Connectors Configuration including Allow anonymous docker pull"*.

Repository Connectors

Connectors allow Docker clients to connect directly to hosted registries, but are not always required. Consult our [documentation](#) for which connector is appropriate for your use case.

HTTP:

Create an HTTP connector at specified port. Normally used if the server is behind a secure proxy.



HTTPS:

Create an HTTPS connector at specified port. Normally used if the server is configured for https.



Allow anonymous docker pull:

Allow anonymous docker pull (Docker Bearer Token Realm required)

Figure: Repository Connectors Configuration including Allow anonymous docker pull

The [Anonymous\(see page 323\)](#) user must be enabled and granted read access to the docker repositories.

- ⓘ Each repository must have the *Allow anonymous docker pull* configuration enabled individually. Enabling this for a group, just allows the anonymous read when utilizing the group connector. If you utilize one of the member connectors, it will use whatever setting it has for that member even if it differs from the group.

Only read settings are affected by this configuration and all other actions on the docker repositories require authentication or lack thereof regardless if this option is on or off.

Accessing Repositories

You can browse Docker repositories in the user interface and inspect the components and assets and their details as documented in [Browsing Repositories and Repository Groups\(see page 227\)](#).

When using the docker command line client, or any other tools using the repository manager indirectly, the common structure for commands can be:

```
docker <command> <nexus-hostname>:<https-repository-port>/<namespace>/<image>:<tag>
docker search <nexus-hostname>:<https-repository-port>/<search-term>
```

with

command

a docker command such as push or pull

nexus-hostname

the IP number or hostname of your repository manager

https-repository-port

https port which ultimately directs to the NXRM docker registry https port in the repository connector config, or that of a reverse proxy https connector which proxies the docker registry plain http port

namespace

the optional namespace of the specific image reflecting the owner, if left out this will silently default to `/library` and utilize Docker Hub

image

the name of the Docker image

tag

the optional tag of the image, defaulting to `latest` when omitted

search-term

the search term or name of the image to search for

The most important aspects are to know and use the correct hostname for the repository manager and the port for the desired repository or repository group.

Searching

Searching for Docker images can be performed in the user interface as described in [Searching for Components](#)(see page 213). This search will find all Docker images that are currently stored in repositories, either because they have been pushed to a hosted repository or they have been proxied from an upstream repository and cached in the repository manager.

The more common use case for a Docker user is to search for images on the command line:

```
$ docker search postgres
NAME      DESCRIPTION                                     STARS  OFFICIAL  AUTOMATED
postgres  The PostgreSQL object-relational database...  1025    [OK]      ...
```

By default this search uses Docker Hub as preconfigured in docker and will only find images available there. A more powerful search is provided by the repository manager when searching against a repository group. An example looking for a `postgres` image on Nexus Repository Manager OSS running on the host `nexus.example.com` and exposing a repository group with a repository connector port of 18443 looks like this:

```
docker search nexus.example.com:18443/postgres
```

The results include all images found in the repositories that are part of the repository group. This includes any private images you have pushed to your hosted repositories. In addition it includes all results returned from the remote repositories configured as proxy repositories in the group. Searching in a specific repository can be achieved by using the repository connector port for the specific repository.

- ⓘ Searching can be done from the command line anonymously by setting up your individual repositories as described in [the earlier subsection](#)(see page 394).

Pulling Images

Downloading images, also known as pulling, from the repository manager can be performed with the docker pull command. The only necessary additions are the hostname or IP address of the repository manager as well as the repository connector port for the repository or repository group to download from:

```
docker pull <nexus-hostname>:<repository-port>/<image>
```

The preferred setup is to proxy all relevant sources of public/private images you want to use, with Docker Hub being the most common choice. Then configure one or more hosted repositories to contain your own images, and expose these repositories through one repository group.

Examples for various images from Nexus Repository Manager running on the host `nexus.example.com` and exposing a repository group with a repository connector port of 18443 are:

```
docker pull nexus.example.com:18443/ubuntu
docker pull nexus.example.com:18443/bitnami/node
docker pull nexus.example.com:18443/postgres:9.4
```

These snippets download the official ubuntu image, the node image from the user bitnami and the version 9.4 of the postgres image. Official images such as ubuntu or postgres belong to the library user on Docker Hub and will therefore show up as `library/ubuntu` and `library/postgres` in the repository manager.

After a successful pull you can start the container with run.

- ⓘ Pulling can be configured to be done from the command line anonymously by setting up your individual repositories as described in [the earlier subsection](#)⁷⁷¹.

⁷⁷¹ <https://help.sonatype.com/display/NXRM3M/Authentication#Authentication-AnonymousReadAccess>

Pushing Images

Sharing an image can be achieved by publishing it to a hosted repository. This is completely private and requires you to tag and push the image. When tagging an image, you can use the image identifier (`imageId`). It is listed when showing the list of all images with `docker images`. Syntax and an example (using `imageId`) for creating a tag are:

```
docker tag <imageId or imageName> <nexus-hostname>:<repository-port>/<image>:<tag>
docker tag af340544ed62 nexus.example.com:18444/hello-world:mytag
```

Once the tag, which can be equivalent to a version, is created successfully, you can confirm its creation with `docker images` and issue the push with the syntax:

```
docker push <nexus-hostname>:<repository-port>/<image>:<tag>
```

⚠ Note that the port needs to be the repository connector port configured for the hosted repository to which you want to push to. You can not push to a repository group or a proxy repository.

A sample output could look like this:

```
$ docker push nexus.example.com:18444/hello-world:labeltest
The push refers to a repository [nexus.example.com:18444/hello-world] (len: 1)
Sending image list
Pushing repository nexus.example.com:18444/hello-world (1 tags)
535020c3e8ad: Image successfully pushed
af340544ed62: Image successfully pushed
Pushing tag for rev [af340544ed62] on
{https://nexus.example.com:18444/repository/docker-internal/v1/repositories/hello-world/tags/labeltest}
```

Now, this updated image is available in the repository manager and can be pulled by anyone with access to the repository, or the repository group, containing the image. Pulling the image from the repository group exposed at port 18443 can be done with:

```
docker pull nexus.example.com:18443/hello-world:labeltest
```

Prior to push, and depending on your configuration, repository manager login credentials may be required before a push or pull can occur.

ⓘ Searching, Browsing, Pushing and Pulling are all showcased in [this video](#)⁷⁷².

Pushing large images can result in failures due to network interruptions and other issues. These partial uploads result in temporary storage for these transfers in the repository manager filling up. The task Purge incomplete docker uploads can be configured to delete these files. If you also tend to upload images to the same tag repeatedly, this can leave a lot of dangling images around, consuming a lot of space. The task Purge unused docker manifests and images can be configured to remove these files. Further information about these tasks can be found in [Configuring and Executing Tasks](#)(see page 258).

Scaling Repositories

When there is a requirement to have many Docker registries it potentially leads to configuring many repositories using many different connector ports. These connector ports are setup within NXRM and use the connector thread allocation logic of the underlying Eclipse Jetty. The default thread pool size, that Eclipse Jetty draws from to preallocate threads for the new connector, is 400. Our support article [Understanding Eclipse Jetty 9.4.8 Thread Allocation](#)⁷⁷³ gives really good details on how allocation works and why at certain amount of repositories with connectors the pool can get maxed out and `IllegalStateException` maybe seen indicating that there are insufficient configured threads.

Reverse Proxy (recommended)

Placing a reverse proxy in front of NXRM dynamically maps docker requests to NXRM Docker repositories. Using a reverse proxy does not require port numbers to be specified and therefore no new connectors on your docker repositories are required. There are two strategies that can be used, the *Port Method* and the *Subdomain Method*. In the support article [Docker Repository Reverse Proxy Strategies](#)⁷⁷⁴ we give more details on these strategies.

Changing the thread pool size

Knowing that there is a limit to the amount of connectors that can be used in conjunction with how threads are being allocated for them from the Eclipse Jetty thread pool gives us a clear indication that we are able to make changes to the configuration of the thread pool. This can be achieved by editing the configuration file (`$install-dir/etc/jetty/jetty.xml`) and add a new `maxThreads` setter (a restart of NXRM is required to pick up changes to the file)

⁷⁷² <https://www.youtube.com/watch?v=Z2jH9Lgeel8>

⁷⁷³ <https://support.sonatype.com/hc/en-us/articles/360000744687>

⁷⁷⁴ <https://support.sonatype.com/hc/en-us/articles/360000761828-Docker-Repository-Reverse-Proxy-Strategies>

```
<Arg name="threadpool">  
    <New id="threadpool" class="org.sonatype.nexus.bootstrap.jetty.InstrumentedQueuedThreadPool">  
        <Set name="maxThreads">400</Set>  
    </New>  
</Arg>
```

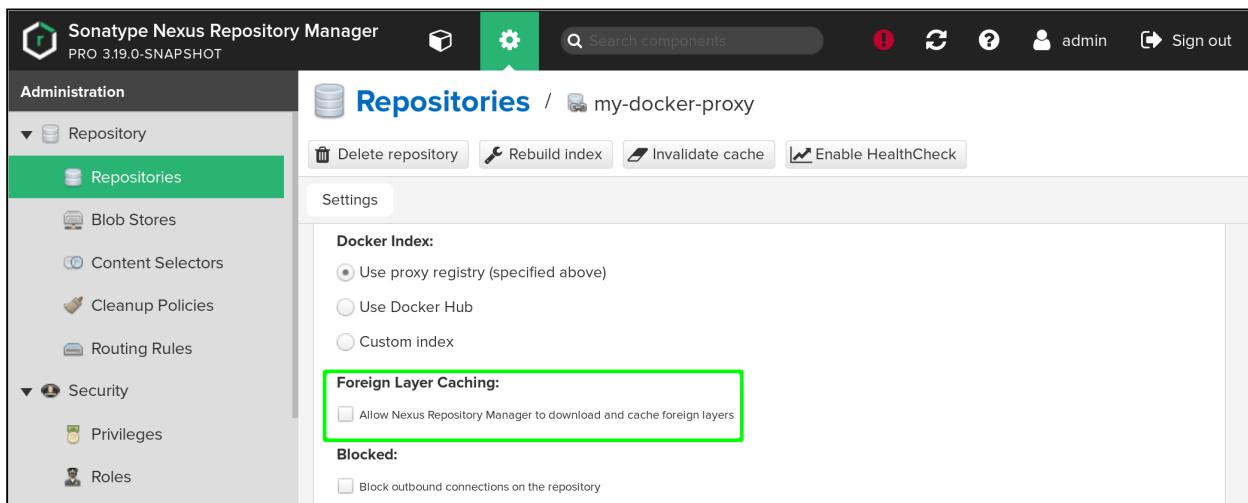
- ⓘ Changing the thread pool size can help with small scaling efficiencies but is not a sustainable solution for scaling performance and we suggest being conservative increasing the maximum threads as each new thread in the pool has the potential to increase workload inside of NXRM for concurrent request threads.

Foreign Layers

NEW IN RELEASE 3.19

You can enable NXRM to cache the foreign layers of Docker images so that the client will not need to communicate to a Docker registry other than your proxied Docker repository. This can be useful in air-gapped environments in which client hosts may not have full access to the internet where the layer may not be accessible.

To enable this functionality, in your Docker proxy repository configuration check the *Foreign Layer Caching* option:



The screenshot shows the Nexus Repository Manager interface. The left sidebar is titled 'Administration' and includes sections for 'Repository' (with 'Repositories' selected), 'Blob Stores', 'Content Selectors', 'Cleanup Policies', 'Routing Rules', 'Security' (with 'Privileges' and 'Roles' listed), and 'Logs'. The main content area is titled 'Repositories / my-docker-proxy'. It features a toolbar with 'Delete repository', 'Rebuild index', 'Invalidate cache', and 'Enable HealthCheck' buttons. Below the toolbar is a 'Settings' tab. Under 'Docker Index:', there are three radio button options: 'Use proxy registry (specified above)' (selected), 'Use Docker Hub', and 'Custom index'. A green box highlights the 'Foreign Layer Caching:' section, which contains a checkbox labeled 'Allow Nexus Repository Manager to download and cache foreign layers'. Another green box highlights the 'Blocked:' section, which contains a checkbox labeled 'Block outbound connections on the repository'.

This will expand the area to display the additional configuration of allowed URL patterns. In order to provide fine grained control over where your proxy repository communicates you will need to add a URL pattern (a regular expression) to the configuration. These patterns act as a white list to determine if your proxy

repository is allowed to retrieve and cache foreign layers from a particular location. The list is inclusive such that a URL that matches **any** configured pattern will be allowed. When caching is enabled a default pattern of `.*` will be added which will match any URL; modify the pattern as desired, and click *Add URL Pattern* to set additional values. At least one value must be set.

Single, Defaulted Entry allowing all URLs:

Foreign Layer Caching:

Allow Nexus Repository Manager to download and cache foreign layers

Foreign Layer Allowed URLs

Regular expressions used to identify URLs that are allowed for foreign layer requests

`.*` Delete

+ Add URL Pattern

Multiple entries:

Foreign Layer Caching:

Allow Nexus Repository Manager to download and cache foreign layers

Foreign Layer Allowed URLs

Regular expressions used to identify URLs that are allowed for foreign layer requests

`https?://go\.microsoft\.com/.*` Delete

`https://.*\.azurecr\.io/.*` Delete

+ Add URL Pattern

For the later example the two patterns would allow foreign layer retrieval from any URL path in the `go.microsoft.com` domain for either http or https protocols, as well as any URL path for any subdomain of the `azurecr.io` domain for only the https protocol.

⚠ It should be noted that the regular expressions used here must match the URL entirely, not just a portion. The examples accomplish this by adding the `.*` to the end of the pattern in order to match any path.

Once setup your docker proxy repository will be able to fetch foreign layer instances, providing a single source of data for the corresponding images.

Docker Content Trust

Docker Content Trust (DCT) allows docker image tags to be cryptographically signed. This allows users to verify the integrity and the publisher of docker data provided by the registry. DCT is enforced at two levels: by the docker client (supported by Docker Community and Enterprise) and by the docker engine (Enterprise only).

Docker Content Trust is not directly handled by NXRM3. You can use Docker Notary in conjunction with NXRM3 to publish and manage trusted Docker content.

Docker Notary Service

Docker Notary can be obtained in binary form the release page on github: <https://github.com/theupdateframework/notary/releases>. Please refer to the [Docker documentation](#)⁷⁷⁵ to install and configure Notary.

Client Configuration

Docker Content Trust is configured by setting the following environment variables:

```
export DOCKER_CONTENT_TRUST=1  
export DOCKER_CONTENT_TRUST_SERVER=https://<notary-server-hostname>
```

With these variables set, docker will enforce content trust and use your notary service to store trust information.

⁷⁷⁵ https://docs.docker.com/notary/getting_started/

- ⓘ The docker client requires a valid HTTPS certificate. If you use a certificate signed by another certificate authority (a self-signed certificate for example), place the certificate in \$HOME/.docker/tls/<notary-server-hostname>/ca.crt

With DCT enabled, docker images will be automatically signed on push. A sample output could look like this:

```
$ docker push nexus.example.com:18079/hello-world:test
The push refers to repository [nexus.example.com:18079/hello-world]
test: digest: sha256:92c7f9c92844bbbb5d0a101b22f7c2a7949e40f8ea90c8b3bc396879d95e899a size: 524
Signing and pushing trust metadata
```

Pulling a docker image by tag will automatically validate the signature. For example:

```
$ docker pull nexus.example.com:18079/hello-world:test-unsigned
Error: remote trust data does not exist for nexus.example.com:18079/hello-world: nexus.example.com:18079
does not have trust data for nexus.example.com:18079/hello-world

$ docker pull nexus.example.com:18079/hello-world:test-unsigned
test: Pulling from hello-world
Digest: sha256:92c7f9c92844bbbb5d0a101b22f7c2a7949e40f8ea90c8b3bc396879d95e899a
Status: Image is up to date for nexus.example.com:18079/hello-world:test
nexus.example.com:18079/hello-world:test
```

Git LFS Repositories

- ⓘ Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

Git Large File Storage (LFS) is a Git extension mechanism that allows large files to be stored outside of a normal Git repository, yet allows end users to interact with those files as if they were part of the same project. Instead of the actual large file, a small pointer file replaces the file in your Git repository, and the file content is stored in a separate location. More information on this format can be found at the [Git LFS homepage](#)⁷⁷⁶.

Nexus Repository Manager Pro and Nexus Repository Manager OSS include the ability to host Git LFS repositories. Nexus Repository Manager's Git LFS implementation supports the batch API, the basic transfer

⁷⁷⁶ <https://git-lfs.github.com/>

adapter for uploading and downloading files, and a place for storing and retrieving the actual files transferred from the Git LFS client. These operations are handled automatically by the Git LFS client once installed on developers' workstations and configured for use with Nexus Repository Manager.

This section details the process of configuring a Git LFS hosted repository and configuring a specific Git project to use Nexus Repository Manager for Git LFS file storage.

Requirements

Git 1.8.2 or above is needed for use of this format.

Only Git LFS versions 1.* and 2.* are supported. 0.* versions have been found not to work.

Creating a Hosted Git LFS Repository

To host Git LFS content in Nexus Repository Manager, you will need to create a new gitlfs (hosted) repository as documented in [Hosted Repository\(see page 0\)](#).

Minimal configuration steps are:

- Define *Name* - e.g. gitlfs-hosted
- Pick a *Blob store for Storage*



Removing content from a Git LFS repository is not recommended and no cleanup task exists.

Removing files from Git LFS means that those files will no longer be available from your associated Git repository, and the pointer files in Git will point to content that no longer exists. A one-to-one mapping between Git LFS repositories and Git repositories and the use of dedicated blobstores may mitigate some concerns by making it easier to add additional storage in the future.

The *Strict Content Type Validation* option has no effect for Git LFS repositories. Since Git LFS by its nature accepts any kind of file content, we perform no content validation on incoming files and assume all files are generic `application/octet-stream` content.

Installing Git LFS Locally

You will need to install Git LFS on your local machine if you have not done so already. Follow the installation directions for Git LFS for your particular platform. After installation, running `git lfs env` is also a good way to confirm that the install succeeded.

Configuring Git LFS Locally

You will need to configure your local Git LFS installation to use Nexus Repository Manager as its Git LFS backend. While we highly recommend the [Git LFS Tutorial](#)⁷⁷⁷, we provide specifics for use with your repository manager.

If you have a Git project that you wish to use with Git LFS and you wish to configure it on a per-project basis, you should use the Git command line tool to configure your `.lfsconfig` file appropriately (making substitutions as appropriate):

```
git config -f .lfsconfig lfs.url http://localhost:8081/repository/gitlfs-hosted/info/lfs
```

Doing so will also create (or modify) your `.lfsconfig` file, which should then be committed and pushed to your Git project so all developers will use the same configuration and Git LFS repository for their large files.

- ⓘ You should carefully consider the mapping between Git LFS hosted repositories and your actual Git projects. The origin of a particular file is not provided to a Git LFS repository, so without such consideration, you will not be able to determine the associated Git repository for a particular file without examining your Git repositories directly.

You also need to tell Git LFS what large files to track. For example:

```
git lfs track '*.*.jpg'
```

This will create (or modify) your `.gitattributes` file, which should be committed and pushed to your Git repository so that it will be shared by all developers.

- ⚠ Do not track your configuration files. If you track them, they will be replaced by pointer files and unexpected behavior may result. In particular, Git LFS may no longer be configured to communicate with Nexus Repository Manager. It is possible under such circumstances that Git LFS will appear to be working correctly but not send or retrieve files from Nexus Repository Manager.

⁷⁷⁷ <https://github.com/git-lfs/git-lfs/wiki/Tutorial>

Go Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

The [Go](#)⁷⁷⁸ programming language introduced [Go modules](#)⁷⁷⁹ starting in Go version 1.11. This feature is intended to make dependency version information explicit and easier to manage.

Nexus Repository Manager OSS and Nexus Repository Manager Pro support Go modules dependency resolution using proxy and group repositories. Go packages can now be cached and served from NXRM using a single endpoint, without builds incurring repeated downloads of packages from the public internet.

 This format is disabled by default in Professional HA-C environments.

The Go programming language natively uses the underlying version control system for fetching dependencies, for example Github.com, whereby the Go command line will use git to perform a full clone of all required dependencies. The Go community realised that this has a number of negative impacts for any typical CI process. Cloning repositories can be a time consuming process. Additionally, fetching sources from any number of version control systems such as Github excludes the ability of a proxying mechanism such as that provided by Nexus Repository Manager. To solve this problem, Go introduced the [GOPROXY](#)⁷⁸⁰ environment variable.

 Learn more on [Go dependency management](#)⁷⁸¹ by reading the technical guide.

⁷⁷⁸ <http://golang.org>

⁷⁷⁹ <https://blog.golang.org/using-go-modules>

⁷⁸⁰ https://research.swtch.com/vgo-module#proxy_servers

⁷⁸¹ https://guides.sonatype.com/repo3/technical-guides/go-dependencies-nxrm3/?utm_source=nxrm3-help-docs&utm_medium=HSC&utm_campaign=guides-promo

GOPROXY

With the introduction of Go modules, the community also introduced an environment variable `$GOPROXY` and, when set, all modules are fetched using a Download API using the base url as the proxying endpoint. The Nexus Repository Manager requires the setting of the `$GOPROXY` environment variable pointing to a Go repository.

Proxying Go Repositories

You can set up a Go proxy repository to access a remote repository location.

- ⓘ At the time of this writing, it must be noted that there is currently no official central repository for Go. It is the client responsibility to choose a repository to proxy, be that public or locally hosted setup, such as using an [Athens server](#)⁷⁸².

To proxy a Go repository, you simply create a new `go (proxy)` as documented in [Repository Management](#)⁷⁸³.

Minimal configuration steps are:

- Define *Name*, e.g. `go-proxy`
- Define URL for *Remote storage*, e.g. `https://gonexus.dev`
- Pick a *Blob store for Storage*

❗ **Non-Anonymous**

At the time of implementation of the Go repository in NXRM, the Go client does not have a means of authentication. This means NXRM instances with anonymous disabled will not be able to reach the Go endpoints, rendering proxy useless. See [here\(see page 323\)](#) for details on anonymous security.

Grouping Go Repositories

A repository group is the recommended way to expose all your Go repositories. A repository group allows you to expose the aggregated content of multiple Go repositories with one URL; it is this url that you should set the `$GOPROXY` environment variable url to.

⁷⁸² <https://docs.gomods.io/>

⁷⁸³ <https://help.sonatype.com/repomanager3/configuration/repository-management>

To create a Go group repository, create a new repository using the recipe *go (group)* as documented in [Repository Management](#)⁷⁸⁴.

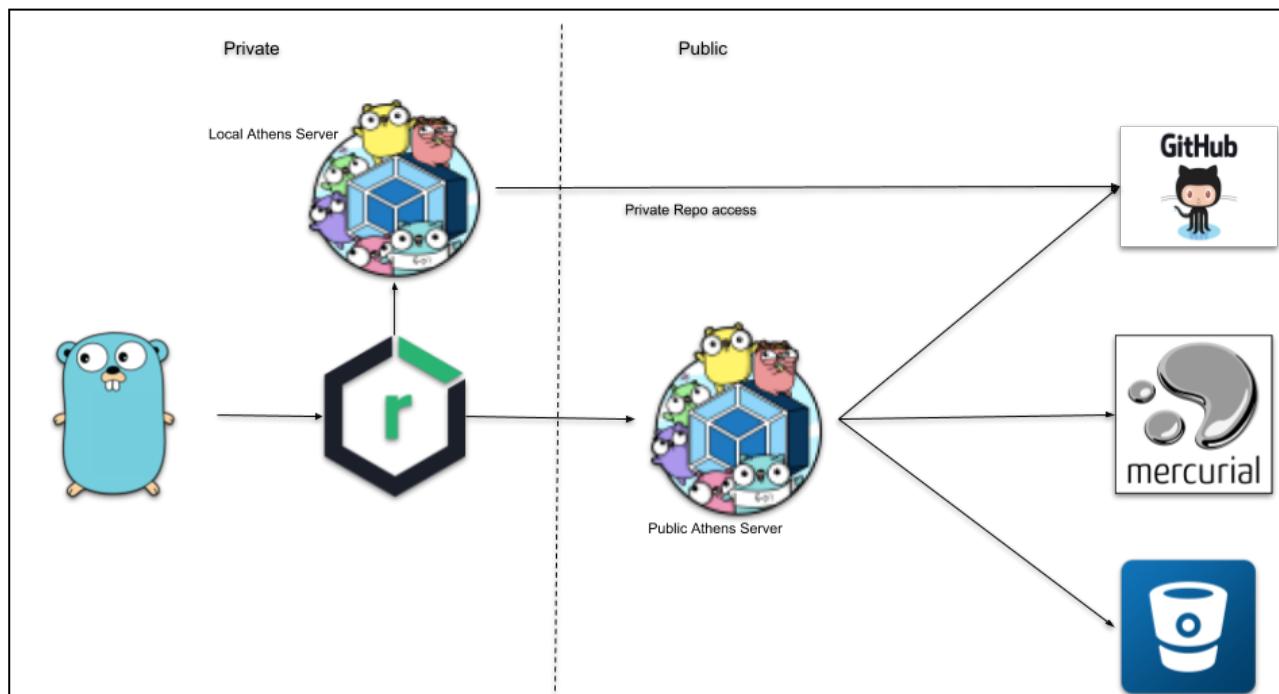
Minimal configuration steps are:

- Define *Name*
- Select *Blob store for Storage*
- Add Go repositories to the *Members* list in the desired order

Go configuration examples

If you need to access dependencies from authenticated sources, you'll need to run [Athens](#)⁷⁸⁵ internally; this locally hosted server is able to be configured with authentication credentials to access private repositories. You may still decide to proxy an external server to avoid unnecessary duplication of caching and therefore storage space.

The following diagram depicts an example set up:



⁷⁸⁴ <https://help.sonatype.com/repomanager3/configuration/repository-management>

⁷⁸⁵ <https://docs.gomods.io/>

Helm Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

NEW IN RELEASE 3.21

Introduction

Helm is the first application package manager running atop Kubernetes(k8s). It allows describing the application structure through convenient helm-charts and managing it with simple commands. More information about helm you can find in the [official documentation](#)⁷⁸⁶.

You can publish your own charts and provenance files to a private repository in a hosted Helm repository.

You can also use a proxy Helm repository reducing time and bandwidth usage for accessing Helm charts as well as more easily share your charts within your organization.

 This format is disabled by default in NXRM Pro HA-C environments.

Proxying Helm Repositories

You can set up a Helm proxy repository to access a remote repository location.

To proxy a Helm repository create a new *helm (proxy)* as shown in the documented example in [Repository Management](#)(see page 250) in detail. Minimal configuration steps for the Helm proxy are:

- Define *Name*, e.g. `helm-proxy`
- Define *URL for Remote storage*, e.g. <https://kubernetes-charts.storage.googleapis.com/>
- Select a *Blob store for Storage*

Hosting Helm Repositories

Creating a Helm hosted repository allows you to register charts in the repository manager. The hosted repository acts as an authoritative location for these components.

⁷⁸⁶ <https://helm.sh/docs/>

To add a hosted helm repository, create a new repository with the recipe `helm (hosted)` as shown in the documented example in [Repository Management](#)(see page 250). Minimal configuration steps for Helm hosted are:

- Define *Name*, e.g. `hosted-hosted`
- Select *Blob store* for *Storage*

Uploading Charts to Helm Hosted Repositories

Upload with Nexus Repository Manager UI

Upload in the UI, as documented [here](#)⁷⁸⁷, to upload charts to helm hosted repository e.g. for all OS, just upload the chart (extension .tgz).

Upload by HTTP POST

The following example uses the curl command to upload `mysql-1.4.0.tgz` file to a hosted Helm repository (fill in <> with your respective data):

```
curl -u <username>:<password> http://<host>:<port>/repository/<repository_name>/ --upload-file  
mysql-1.4.0.tgz -v
```

Uploading Provenance Files to Helm Hosted Repositories

Uploading provenance files with extension `.prov.tgz` to a hosted repository is the same as uploading a chart. The provenance file will be at the same tree structure level as the chart.

More information about provenance files you can get from the official documentation [here](#)⁷⁸⁸.

Configuring Helm Client

More information you be gotten from the official documentation [here](#)⁷⁸⁹.

Once you have Helm up and running you'll want to run a command similar to the following to add a Helm repo:

⁷⁸⁷ <https://help.sonatype.com/repomanager3/user-interface/uploading-components>

⁷⁸⁸ <https://helm.sh/docs/topics/provenance/>

⁷⁸⁹ <https://helm.sh/docs/intro/install/>

```
helm repo add <helm_repository_name> http://<host>:<port>/repository/<nexus_repository_name>/ --username <username> --password <password>
```

The below command will fetch the latest chart or with the version:

1. helm fetch <helm_repository_name>/<chart_name>
2. helm fetch <helm_repository_name>/<chart_name> --version <chart_version>

For example, NXRM has Helm proxy repository called `helm-proxy` and your NXRM is running on `localhost:8081` where username is `admin` and password is `admin`. You would like to add this repository to Helm client. Also, you would like to fetch the latest MySQL chart.

1. helm repo add nexusrepo http://localhost:8081/repository/helm-proxy/ --username admin --password admin
2. helm fetch nexusrepo/mysql

If you want to fetch a chart with a specific version, just run it, like so:

```
helm fetch nexusrepo/mysql --version 1.4.0
```

Maven Repositories

Introduction

Historically Nexus Repository Manager started as a repository manager supporting the Maven repository format and it continues to include excellent support for users of Apache Maven, Apache Ant/Ivy, Eclipse Aether, Gradle and others.

This section explains the default configuration included in Nexus Repository Manager Pro and Nexus Repository Manager OSS, instructions for creating further Maven repositories as well as searching and browsing the repositories. Build tool configuration for Apache Maven, Apache Ant, Gradle and others tools follow. The configuration examples take advantage of the repository manager merging many repositories and exposing them via a repository group.

- ⓘ Sonatype offers additional education on Maven repository configuration. Go to the [learning page⁷⁹⁰](#) to try our free, self-paced course.

⁷⁹⁰ https://help.sonatype.com/learning/repository-manager-3/first-time-installation-and-setup?utm_campaign=course-promo&utm_medium=help&utm_source=m2

Maven Repository Format

Apache Maven⁷⁹¹ created the most widely used repository format in the Java development ecosystem with the release of Apache Maven 2. It is used by all newer versions of Apache Maven and many other tools including Apache Ivy, Gradle, sbt, Eclipse Aether and Leiningen. Further information about the format can be found in [An Example - Maven Repository Format](#)(see page 163).

The format is used by many publicly available repositories. The [Central Repository](#)⁷⁹² is the largest repository of components aimed at Java/JVM-based development and beyond and is used the Maven repository format for release components of numerous open source projects. It is configured as a proxy repository by default in Apache Maven and widely used in other tools.

In addition to the generic repository management features documented in [Repository Management](#)(see page 234), specifics of the Maven repository format can be configured for each repository in the [Maven 2](#) section:

Version policy

Release

A Maven repository can be configured to be suitable for release components with the Release version policy. The Central Repository uses a release version policy.

Snapshot

Continuous development is typically performed with snapshot versions supported by the Snapshot version policy. These version values have to end with `-SNAPSHOT` in the POM file. This allows repeated uploads where the actual number used is composed of a date/timestamp and an enumerator and the retrieval can still use the `-SNAPSHOT` version string. The repository manager and client tools manage the metadata files that manage this translation from the snapshot version to the timestamp value.

Mixed

The `Mixed` version policy allows you to support both approaches within one repository.

Layout policy

The Maven repository format defines a directory structure as well as a naming convention for the files within the structure. Apache Maven follows these conventions. Other build tools, such as sbt, and custom tools have historically created usages that use the directory structure less strictly, violating the file naming conventions. Components based on these tools' different conventions have been published to public repositories, such as the Central Repository. These tools rely on these changed conventions.

Permissive

⁷⁹¹ <http://maven.apache.org/>

⁷⁹² <http://central.sonatype.org/>

You can configure a layout policy of *Permissive* to allow assets in the repository that violate the default format.

Strict

The default value of *Strict* requires publishing and accessing tools to follow the Apache Maven conventions. This is the preferred setting if you are using Apache Maven, Eclipse Aether, and other strictly compatible tools.

Proxying Maven Repositories

A default installation of Nexus Repository Manager includes a proxy repository configured to access the Central Repository via HTTPS using the URL `https://repo1.maven.org/maven2/`. To reduce duplicate downloads and improve download speeds for your developers and CI servers, you should proxy all other remote repositories you access as proxy repositories as well.

To proxy a Maven repository, you simply create a new repository using the recipe *maven2 (proxy)* as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Define URL for *Remote storage* e.g. `https://repo1.maven.org/maven2/`
- Select a *Blob store for Storage*

This creates a repository using the *Release* version policy and a *Strict* layout policy. Both can be configured as appropriate for the remote repository.

If the remote repository contains a mixture of release and snapshot versions, you have to set the version policy to *Mixed*.

Usage of the repository with build tools such as sbt, potentially requires the layout policy to be set to *Permissive*.

Proxying the Oracle Maven Repository

Proxying the [Oracle Maven Repository](#)⁷⁹³ requires special HTTP options for the *maven2 (proxy)* recipe. Also, you must register for an account to access the external repository. Configure the proxy repository to access the Oracle repository, with these additional steps:

1. Add `https://maven.oracle.com` to the *Remote storage* field.
2. Check *Authentication*, in the *HTTP* section.
3. Enter the *Username* and *Password* from your Oracle account.
4. Check *HTTP request settings*.
5. Check *Enable circular redirects*.

⁷⁹³ https://docs.oracle.com/middleware/1213/core/MAVEN/config_maven_repo.htm#MAVEN9010

6. Check *Enable cookies*.

After applying these settings to your repository, data requests will be redirected to a queue of different URLs, most of which are involved with authentication. By enabling these options, you allow the repository manager to maintain the authentication state in a cookie that would be sent with each request, eliminating the need for the authentication-related redirects and avoiding timeouts.

Hosting Maven Repositories

A hosted Maven repository can be used to deploy your own as well as third-party components. A default installation of Nexus Repository Manager includes two hosted Maven repositories. The *maven-releases* repository uses a release version policy and the *maven-snapshots* repository uses a snapshot version policy.

To create another hosted Maven repository, add a new repository with the recipe *maven2 (hosted)* as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Select *Blob store for Storage*

Grouping Maven Repositories

A repository group is the recommended way to expose all your Maven repositories from the repository manager to your users, without needing any further client side configuration. A repository group allows you to expose the aggregated content of multiple proxy and hosted repositories as well as other repository groups with one URL for tool configuration. This is possible for Maven repositories by creating a new repository with the *maven2 (group)* recipe as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Select *Blob store for Storage*
- Add Maven repositories to the *Members* list in the desired order

A typical, useful example is the *maven-public* group that is configured by default. It aggregates the *maven-central* proxy repository with the *maven-releases* and *maven-snapshots* hosted repositories. Using the URL of the repository group gives you access to the packages in all three repositories with one URL. Any new component added as well as any new repositories added to the group will automatically be available.

Browsing and Searching Maven Repositories

You can browse Maven repositories in the user interface inspecting the components and assets and their details as documented in [Browsing Repositories and Repository Groups](#)(see page 227).

Components can be searched in the user interface as described in [Searching for Components\(see page 213\)](#). A search finds all components and assets that are currently stored in the repository manager, either because they have been deployed to a hosted repository or they have been proxied from an upstream repository and cached in the repository manager.

- ✓ You can change the default column order in the search and browse user interfaces to the familiar order of *Group* (`groupId`), *Name* (`artifactId`) and *Version*. Simply drag the *Group* column from the middle to the left using the header. This setting will be persisted as your preference in your web browser.

Configuring Apache Maven

To use repository manager with [Apache Maven](#)⁷⁹⁴, configure Maven to check the repository manager instead of the default, built-in connection to the Central Repository.

To do this, you add a mirror configuration and override the default configuration for the central repository in your `~/.m2/settings.xml`, shown below:

⁷⁹⁴ <http://maven.apache.org/>

Configuring Maven to Use a Single Repository Group

```
<settings>
  <mirrors>
    <mirror>
      <!--This sends everything else to /public -->
      <id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://localhost:8081/repository/maven-public/</url>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>nexus</id>
      <!--Enable snapshots for the built in central repo to direct -->
      <!--all requests to nexus via the mirror -->
      <repositories>
        <repository>
          <id>central</id>
          <url>http://central</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>http://central</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
  <activeProfiles>
    <!--make the profile active all the time -->
    <activeProfile>nexus</activeProfile>
  </activeProfiles>
</settings>
```

In *Configuring Maven to Use a Single Repository Group* a single profile called `nexus` is defined. It configures a repository and a pluginRepository with the id `central` that overrides the same repositories in the Super POM. The Super POM is internal to every Apache Maven install and establishes default values. These overrides are important since they change the repositories by enabling snapshots and replacing the URL with a bogus URL. This URL is overridden by the `mirror` setting in the same `settings.xml` file to point to the URL of your single repository group. This repository group can, therefore, contain release as well as snapshot components and Maven will pick them up.

The `mirrorOf` pattern of `*` causes any repository request to be redirected to this mirror and to your single repository group, which in the example is the public group.

It is possible to use other patterns in the `mirrorOf` field. A possible valuable setting is to use `external:*`. This matches all repositories except those using `localhost` or file based repositories. This is used in conjunction with a repository manager when you want to exclude redirecting repositories that are defined for integration testing. The integration test runs for Apache Maven itself require this setting.

More documentation about mirror settings can be found in the [mini guide on the Maven web site](#)⁷⁹⁵.

As a last configuration the `nexus` profile is listed as an active profile in the `activeProfiles` element.

Deployment to a repository is configured in the `pom.xml` for the respective project in the `distributionManagement` section. Using the default repositories of the repository manager:

```
<project>
...
<distributionManagement>
    <repository>
        <id>nexus</id>
        <name>Releases</name>
        <url>http://localhost:8081/repository/maven-releases</url>
    </repository>
    <snapshotRepository>
        <id>nexus</id>
        <name>Snapshot</name>
        <url>http://localhost:8081/repository/maven-snapshots</url>
    </snapshotRepository>
</distributionManagement>
...
```

The credentials used for the deployment are found in the `server` section of your `settings.xml`. In the example below `server` contains `nexus` as the `id`, along with the default username and password:

```
<settings>
...
<servers>
    <server>
        <id>nexus</id>
        <username>admin</username>
        <password>admin123</password>
    </server>
</servers>
```

Full example projects can be found in the `maven` folder of the [example project](#)⁷⁹⁶ in the `nexus-3.x` branch. A full build of the `simple-project`, including downloading the declared dependencies and uploading the build output to the repository manager can be invoked with `mvn clean deploy`.

⁷⁹⁵ <http://maven.apache.org/guides/mini/guide-mirror-settings.html>

⁷⁹⁶ <https://github.com/sonatype/nexus-book-examples>

Configuring Apache Ant and Apache Ivy

Apache Ivy⁷⁹⁷ is a dependency manager often used in Apache Ant builds. It supports the Maven repository format and can be configured to download dependencies that can be declared in the `ivy.xml` file. This configuration can be contained in the `ivysettings.xml`. A minimal example for resolving dependencies from a repository manager running on localhost is shown below:

Minimal Ivy Configuration in an Ant file

```
<ivysettings>
  <settings defaultResolver="nexus"/>
  <property name="nexus-public"
    value="http://localhost:8081/repository/maven-public/" />
  <resolvers>
    <ibiblio name="nexus" m2compatible="true" root="${nexus-public}" />
  </resolvers>
</ivysettings>
```

These minimal settings allow the `ivy:retrieve` task to download the declared dependencies.

To deploy build outputs to a repository with the `ivy:publish` task, user credentials and the URL of the target repository have to be added to `ivysettings.xml` and the `makepom` and `publish` tasks have to be configured and invoked.

Full example projects can be found in the `ant-ivy` folder of the [example project](#)⁷⁹⁸ in the `nexus-3.x` branch. A full build of the `simple-project`, including downloading the declared dependencies and uploading the build output to the repository manager can be invoked with:

```
cd ant-ivy/simple-project
ant deploy
```

Configuring Apache Ant and Eclipse Aether

Eclipse Aether⁷⁹⁹ is the dependency management component used in Apache Maven 3+. The project provides Ant tasks that can be configured to download dependencies that can be declared in a `pom.xml` file or in the Ant build file directly.

This configuration can be contained in your Ant `build.xml` or a separate file that is imported. A minimal example for resolving dependencies from a repository manager running on localhost is shown below:

⁷⁹⁷ <http://ant.apache.org/ivy>

⁷⁹⁸ <https://github.com/sonatype/nexus-book-examples/tree/nexus-3.x>

⁷⁹⁹ <https://projects.eclipse.org/projects/technology.aether>

Minimal Aether Configuration in an Ant file

```
<project xmlns:aether="antlib:org.eclipse.aether.ant" ....>
  <taskdef uri="antlib:org.eclipse.aether.ant" resource="org/eclipse/aether/ant/antlib.xml">
    <classpath>
      <fileset dir="${aether.basedir}" includes="aether-ant-tasks-*.jar" />
    </classpath>
  </taskdef>
  <aether:mirror id="mirror" url="http://localhost:8081/repository/maven-public/" mirrorOf="*"/>
...
</project>
```

These minimal settings allow the `aether:resolve` task to download the declared dependencies.

To deploy build outputs to a repository with the `aether:deploy` task, user authentication and details about the target repositories have to be added.

Full example projects can be found in the `ant-aether` folder of the [example project](#)⁸⁰⁰ in the `nexus-3.x` branch. A full build of the `simple-project`, including downloading the declared dependencies and uploading the build output to the repository manager can be invoked with:

```
cd ant-aether/simple-project
ant deploy
```

Configuring Gradle

[Gradle](#)⁸⁰¹ has a built in dependency management component that supports the Maven repository format. In order to configure a Gradle project to resolve dependencies declared in the `build.gradle` file, a Maven repository as shown in *Gradle Repositories Configuration* has to be declared.

Gradle Repositories Configuration

```
repositories {
  maven {
    url "http://localhost:8081/repository/maven-public/"
  }
}
```

These minimal settings allow Gradle to download the declared dependencies.

To deploy build outputs to a repository with the `uploadArchives` task, user authentication can be declared in e.g., `gradle.properties`:

⁸⁰⁰ <https://github.com/sonatype/nexus-book-examples/tree/nexus-3.x>

⁸⁰¹ <http://www.gradle.org/>

```
nexusUrl=http://localhost:8081  
nexusUsername=admin  
nexusPassword=admin123
```

Then, the authentication values can be used in the `uploadArchives` task with a `mavenDeployer` configuration from the Maven plugin:

```
uploadArchives {  
    repositories {  
        mavenDeployer {  
            repository(url: "${nexusUrl}/repository/maven-releases/") {  
                authentication(userName: nexusUsername, password: nexusPassword)  
            }  
            snapshotRepository(url: "${nexusUrl}/repository/maven-snapshots") {  
                authentication(userName: nexusUsername, password: nexusPassword)  
            }  
        }  
    }  
}
```

Full example projects can be found in the gradle folder of the [examples project](#)⁸⁰² in the `nexus-3.x` branch. A full build of the `simple-project`, including downloading the declared dependencies and uploading the build output to the repository manager can be invoked with:

```
cd gradle/simple-project  
gradle upload
```

SBT

[sbt](#)⁸⁰³ has a built in dependency management component and defaults to the Maven repository format. In order to configure a sbt project to resolve dependencies declared in `build.sbt` file, a resolver, as shown in *SBT Resolvers Configuration* has to be declared.

SBT Resolvers Configuration

```
resolvers += "Nexus" at "http://localhost:8081/repository/maven-public/"
```

These minimal settings allow sbt to download the declared dependencies.

To deploy build outputs to a repository with the `publish` task, user credentials can be declared in the `build.sbt` file:

⁸⁰² <https://github.com/sonatype/nexus-book-examples/tree/nexus-3.x>

⁸⁰³ <http://www.scala-sbt.org/>

```
credentials += Credentials("Sonatype Nexus",
  "nexus.scala-tools.org", "admin", "admin123")
```

The publishTo configuration:

```
publishTo <=> version { v: String =>
  val nexus = "http://localhost:8081/"
  if (v.trim.endsWith("SNAPSHOT"))
    Some("snapshots" at nexus + "repository/maven-snapshots")
  else
    Some("releases" at nexus + "repository/maven-releases")
```

Further documentation can be found in the [sbt documentation on publishing](#)⁸⁰⁴.

Leiningen

[Leiningen](#)⁸⁰⁵ has a built in dependency management component and defaults to the Maven repository format. As a build tool it is mostly used for projects using the Clojure language. Many libraries useful for these projects are published to the Clojars repository. If you want to use these, you have to create two proxy repositories with the remote URL <http://clojars.org/repo/>.

This repository is mixed and you therefore have to create a release and a snapshot proxy repository and then add both to the public group.

In order to configure a Leiningen project to resolve dependencies declared in the `project.clj` file, a `mirrors` section overriding the built in `central` and `clojars` repositories as shown in *Leiningen Configuration* has to be declared.

Leiningen Configuration

These minimal settings allow Leiningen to download the declared dependencies.

```
:mirrors {
  "central" {:name "Nexus"
    :url "http://localhost:8081/repository/maven-public/"
    :repo-manager true}
  #clojars {:name "Nexus"
    :url "http://localhost:8081/repository/maven-public/"
    :repo-manager true}
}
```

To deploy build outputs to a repository with the `deploy` command, the target repositories have to be add to `project.clj` as `deploy-repositories`. This avoids Leiningen checking for dependencies in these

⁸⁰⁴ <http://www.scala-sbt.org/release/docs/Publishing.html>

⁸⁰⁵ <http://leinigen.org/>

repositories, which is not necessary, since they are already part of the public repository group used in mirrors.

```
:deploy-repositories [
  ["snapshots" "http://localhost:8081/repository/maven-snapshots"]
  ["releases" "http://localhost:8081/repository/maven-releases"]
]
```

User credentials can be declared in `~/.lein/credentials.clj.gpg` or will be prompted for.

Further documentation can be found on the [Leiningen website](#)⁸⁰⁶.

npm Registry

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

The command line tool `npm` is a package management solution for Javascript-based development. It is used to create and use node packaged modules and is built into the Javascript platform [Node.js](#)⁸⁰⁷.

NXRM supports the npm registry format for proxy repositories. This allows you to take advantage of the packages in the npm registry and other public registries without incurring repeated downloads of packages, since they will be proxied in the repository manager.

In addition, Nexus Repository Manager supports running your own private registry, also known as a hosted repository, using the `npm` format. You can share internally developed, proprietary packages within your organization via these private registries allowing you to collaborate efficiently across development teams with a central package exchange and storage location.

To share a package or tool with npm, you create a npm package and store it in the npm registry hosted by the repository manager. Similarly, you can use packages others have created and made available in their npm repositories by proxying them or downloading the packages and installing them in your own private registry for third party packages.

To simplify configuration Nexus Repository Manager supports aggregation of npm registries into a single group. This allows you to expose all the external packages from the npm registry and other public registries as well as the private registries as one registry, which greatly simplifies client configuration.

⁸⁰⁶ <http://leinigen.org/>

⁸⁰⁷ <http://www.nodejs.org/>

Proxying npm Registries

To reduce duplicate downloads and improve download speeds for your developers and CI servers, you should proxy the registry served at <https://registry.npmjs.org>. By default npm accesses this registry directly. You can also proxy any other registries you require.

To proxy an external npm registry, you simply create a new *npm (proxy)* as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Define URL for Remote storage e.g. <https://registry.npmjs.org>
- Select *Blob store* for Storage

Private npm Registries

A private npm registry can be used to upload your own packages as well as third-party packages. You can create a private npm registry by setting up a hosted repository with the *npm* format in the repository manager. It is good practice to create two separate hosted repositories for these purposes.

To create a hosted repository with npm format, simply create a new *npm (hosted)* as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Select *Blob store* for Storage

The npm registry information is immediately updated as packages are deployed or deleted from the repository.

Grouping npm Registries

A repository group is the recommended way to expose all your npm registries repositories from the repository manager to your users, with minimal client side configuration. A repository group allows you to expose the aggregated content of multiple proxy and hosted repositories with one URL to npm and other tools. This is possible for npm repositories by creating a new *npm (group)* as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Select *Blob store* for Storage
- Add npm repositories to the *Members* list in the desired order

A typical, useful example would be to group a npm hosted repository with internal software packages, another npm hosted repository with third-party packages and the proxy repository that proxies the npm registry.

Using the URL of the repository group as your npm repository URL in your client tool will give you access to the packages in all three repositories with one URL. Any new packages added as well as any new repositories added to the group will automatically be available.

Browsing npm Registries and Searching Modules

You can browse npm repositories in the user interface inspecting the components and assets and their details as documented in [Browsing Repositories and Repository Groups](#)(see page 227).

Searching for npm modules can be performed in the user interface as described in [Searching for Components](#)(see page 213). This search will find all npm modules images that are currently stored in the repository manager, either because they have been pushed to a hosted repository or they have been proxied from an upstream repository and cached in the repository manager.

Configuring npm

Once you have set up your hosted and proxy repositories for npm packages, and created a repository group to merge them, you can access them with the npm tool on the command line as one registry.

You can configure the registry used by npm in your `.npmrc` file located in your user's home directory with the npm config command and the public URL of your repository group available in the repository list by clicking the *Copy* button in the *URL* column.

For example:

```
npm config set registry http://localhost:8081/repository/npm-all/
```

The command inserts the configuration in the `.npmrc` file in your users home directory.

Registry configuration in `.npmrc`

```
registry = http://localhost:8081/repository/npm-all/
```

With this configuration any npm commands will use the new registry from the repository manager. The command line output will reference the URLs in --verbose mode or with info logging for the downloads of the requested packages:

```
$ npm --loglevel info install grunt
...
npm http fetch GET http://localhost:8081/repository/npmjs-org/grunt/-/grunt-0.4.5.tgz
npm http fetch 200 http://localhost:8081/repository/npmjs-org/grunt/-/grunt-0.4.5.tgz
...
npm http fetch GET http://localhost:8081/repository/npm-all/underscore/-/underscore-1.7.0.tgz
npm http fetch 200 http://localhost:8081/repository/npm-all/underscore/-/underscore-1.7.0.tgz
...
```

npm Security

If enabled, any anonymous user has read access to the repositories and repository groups. If anonymous access, as documented in [Anonymous Access](#)(see page 323), is disabled or write access is required for publishing a package, the user needs to authenticate to the repository manager. There are two methods to authenticate npm with your repository manager. Only one should be used at a time.

Authentication Using Realm and Login

This authentication method requires the *npm Bearer Token Realm*. Simply add the realm to the active realms in the Realms feature of the Security menu from the Administration menu to activate it as documented in [Realms](#)(see page 0).

Once the realm is activated, a npm CLI user can establish the authentication to a repository with the `npm adduser`⁸⁰⁸ (`npm login` is an equivalent alias) command.

```
npm adduser --registry=http://localhost:8081/repository/npm-internal/
```

Provide your repository manager username and password as well as an email address when prompted. Upon successful completion, a line for authentication of this combination is automatically added to your `.npmrc` configuration file for the specific repository.

Despite its name, the `npm adduser` command does not actually create a user account inside NXRM. It merely associates a token with an existing user account and allows the CLI to store that token for re-use.

Authentication Using Basic Auth

In some instances you cannot use the realm and login method, for example if you have a username which includes capital letters (disallowed by `npm login`). In these you can still use npm by configuring it to use basic auth with your repository manager. This authentication method involves editing the `.npmrc` configuration file adding an encoded username and password as well as configuring authentication to always occur. It is considered the less flexible of the methods supported.

⁸⁰⁸ <https://docs.npmjs.com/cli-commands/adduser.html>

The `_auth` variable has to be generated by base64-encoding the string of `username:password`. You can create this encoded string with the command line call to `openssl` e.g. for the admin user using password `admin123`:

```
echo -n 'admin:admin123' | openssl base64
```

Other tools for the encoding are `uuencode` or, for Windows users, `certutil`. To use `certutil` on Windows you need to put the credentials to be encoded into a file:

```
admin:admin123
```

- ⓘ Ensure your file does not have extra whitespace or a trailing line separator as either of these will negatively impact the resultant output.

Then run:

```
c:\certutil /encode in.txt out.txt
```

After this the base64 encoded credentials can be found in between the begin and end certificate lines in the output file:

```
-----BEGIN CERTIFICATE-----  
YWRtaW46YWRtaW4xMjM=  
-----END CERTIFICATE-----
```

- ⓘ Whatever tool you use to generate the encoded username and password string can be tested by encoding the string `admin:admin123`, which should result in `YWRtaW46YWRtaW4xMjM=`. Another example is `jane:testpassword123` which should result in `amFuZTp0ZXN0cGFzc3dvcmQxMjM=`.

Once you have encoded credentials they can be added to the `.npmrc` file, along with your author email and enabled authentication (below your already entered registry configuration). For example, for `admin/admin123`:

```
email=you@example.com  
always-auth=true  
_auth=YWRtaW46YWRtaW4xMjM=
```

With one (not both) of these authentication methods in place, you are ready to publish.

Publishing npm Packages

Publishing your own packages to a npm hosted repository allows you to share packages across your organization or with external partners. With authentication configured you can publish your packages with the `npm publish` command.

The `npm publish` command uses a registry configuration value to know where to publish your package. There are several ways to change the registry value to point at your hosted npm repository.

Since the `.npmrc` file usually contains a registry value intended only for getting new packages, a simple way to override this value is to provide a registry to the publish command:

```
npm publish --registry http://localhost:8081/repository/npm-internal/
```

Alternately, you can edit your `package.json` file and add a `publishConfig` section:

```
"publishConfig" : {  
  "registry" : "http://localhost:8081/repository/npm-internal/"  
},
```

Detailed information about package creation can be found on the [npm website](#)⁸⁰⁹.

If your package requires the use of `npm scope`, the repository manager supports this functionality. Packages published to the repository manager with a defined scope are reflected with the scope value populating the repository group field in Browse and Search. Details on scoping are available on the [npm website](#)⁸¹⁰ also.

Once a package is published to the private registry in the repository manager, any other developers or build servers that access the repository manager via the repository group have instant access to it.

Deprecating npm Packages

Once your packages have been pushed to an npm hosted repository, you can mark them as deprecated. This is useful when a newer version of the package is available and you want to warn people that the old package has reached end of life, or you want to avoid usage and warn your users for some other reason.

The `npm deprecate` command uses a `registry` configuration value to inform where the package lives. To deprecate an existing package, use a command like the following:

```
npm deprecate --registry http://localhost:8081/repository/npm-internal/ testproject1@0.0.1 "This package is  
deprecated"
```

⁸⁰⁹ <https://docs.npmjs.com/cli/publish>

⁸¹⁰ <https://docs.npmjs.com/getting-started/scoped-packages>

If you change your mind, you can reverse this action using the same command. To un-deprecate a package, pass an empty string to the deprecate command:

```
npm deprecate --registry http://localhost:8081/repository/npm-internal/ testproject1@0.0.1 ""
```

The message text is persisted in the deprecated attribute of the packageJson section for the asset and can be viewed in the user interface.

NuGet Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

With the creation of the NuGet project, a package management solution for .NET developers has become available. Similar to Apache Maven dependency management for Java developers, NuGet makes it easy to add, remove and update libraries and tools in Visual Studio projects that use the .NET Framework.

The project website at www.nuget.org⁸¹¹ hosts tool downloads and detailed documentation as well as links to further resources and provide a repository and features to upload your open source NuGet packages. With the NuGet Gallery, a repository of open source libraries and tools is available and the need for repository management arises.

Nexus Repository Manager Pro and Nexus Repository Manager OSS support the NuGet repository format for hosted and proxy repositories as well as exposing them to the client-side tools as a repository group and have related repositories preconfigured.

Nexus Repository Manager and NuGet allow you to improve collaboration and control, while speeding up .NET development, by facilitating open source libraries and sharing of internal component across teams. When you standardize on a single repository for all your development and use it for internal components, as well, you will get all the benefits of using a repository manager when working in the .NET architecture.

To share a library or tool with NuGet, you create a NuGet package and store it in the NuGet repository on the repository manager. Similarly, you can use packages others have created and made available in their NuGet repositories by proxying them or downloading the packages and pushing them to your own hosted repository for third party packages.

NuGet Repository Format

The NuGet repository format uses OData queries for communication between the client and the repository. These queries include metadata information about available packages and other data.

⁸¹¹ <http://www.nuget.org>

When the repository manager receives queries from the NuGet client, it passes these queries on to the remote repositories. To avoid sending identical queries to the remote repository, the repository manager caches the queries and will rely on previously stored metadata if the same query is received again before the cache expires.

The *NuGet* section is included in NuGet proxy repositories to allow configuration of this caching. Specifically, the parameter *Metadata query cache age* can be used to configure the rate at which queries expire and are subsequently re-run.

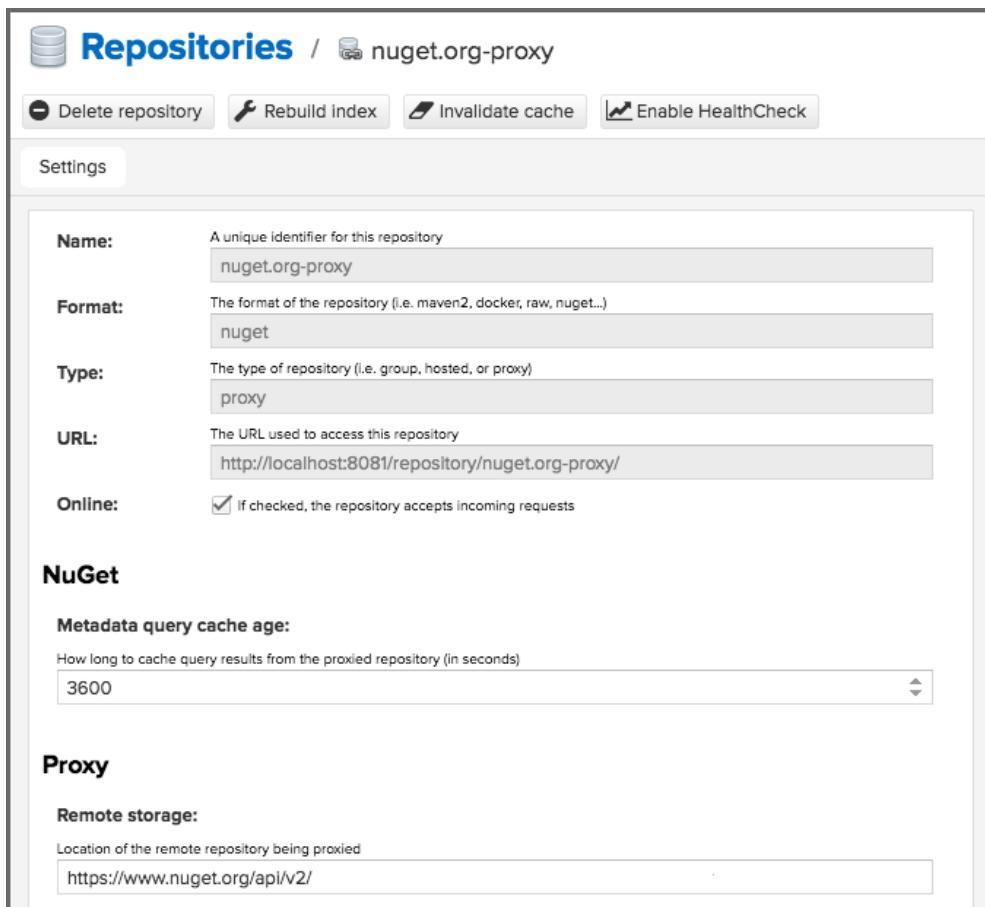
Topics in this section:

- [NuGet Proxy Repositories](#)(see page 429)
- [NuGet Hosted Repositories](#)(see page 432)
- [Grouping NuGet Repositories](#)(see page 433)
- [Accessing Packages in Repositories and Groups](#)(see page 434)
- [Deploying Packages to NuGet Hosted Repositories](#)(see page 434)
- [Integration with Visual Studio](#)(see page 435)

NuGet Proxy Repositories

The NuGet Gallery is the common repository used by all package authors and consumers. To reduce duplicate downloads and improve download speeds for your developers and CI servers, you should proxy the NuGet Gallery with the repository manager. If you use other external repositories, you should also proxy these. A default installation of the repository manager has the NuGet gallery set up as a proxy repository with the name *nuget.org-proxy*.

To proxy another external NuGet repository, you simply create a new nuget (proxy) as documented in [Repository Management](#)(see page 234). The Remote Storage has to be set to the URL of the remote repository you want to proxy. The default configuration for proxying the NuGet Gallery is partially visible in *Figure: “NuGet Proxy Repository Configuration for the NuGet Gallery”* and uses the URL of the API <https://www.nuget.org/api/v2/>.



The screenshot shows the 'Repositories' section of the Nexus Repository Manager 3 interface. A specific repository named 'nuget.org-proxy' is selected. The configuration details are as follows:

- Name:** nuget.org-proxy
- Format:** nuget
- Type:** proxy
- URL:** http://localhost:8081/repository/nuget.org-proxy/
- Online:** checked (repository accepts incoming requests)

NuGet section settings:

- Metadata query cache age:** 3600 seconds

Proxy section settings:

- Remote storage:** https://www.nuget.org/api/v2/

Figure: NuGet Proxy Repository Configuration for the NuGet Gallery

By default, searches in NuGet proxy repositories in the repository manager initiated by a client like nuget or VisualStudio will be passed through to the remote repositories. The search results are merged with internal search results and included in an internally managed index. This merging has to make some assumptions to generate component counts. These counts should therefore be considered approximate numbers. The cache can be configured in the NuGet section documented in [NuGet Repository Format⁸¹²](#).

NuGet Server API Version 3 Support

NEW IN RELEASE 3.21

The V3 API for NuGet is supported by NXRM NuGet Proxy repositories. Read more about the API at <https://docs.microsoft.com/en-us/nuget/api/overview>. NuGet.org has updated its package service to version 3 which provides significant performance improvements over its legacy version 2 (OData-based) service.

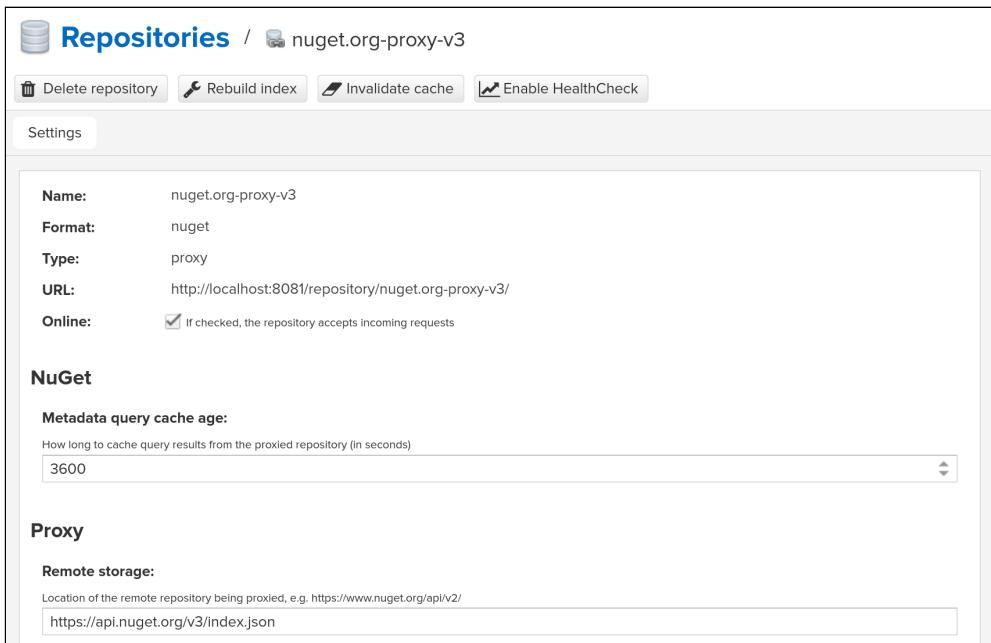
⁸¹ <https://help.sonatype.com/display/>

NXRM3/.NET+Package+Repositories+with+NuGet#id-.NETPackageRepositorieswithNuGet-NugetRepositoryFormat

⚠ NXRM currently supports the V3 API **only** for proxy repositories. Group and hosted support will be added in a subsequent release.

If a NuGet Proxy Repository is configured to a V3 remote and is a member of a group it will not be able to participate in group requests and potentially cause build errors.

In order to leverage V3 behavior the Remote Storage property of the repository must be set to the [Service Index](#)⁸¹³ of the upstream target. For nuget.org this means setting the value to <https://api.nuget.org/v3/index.json>.



The screenshot shows the 'Repositories' page in Nexus Repository Manager. A specific repository named 'nuget.org-proxy-v3' is selected. The configuration details are as follows:

- Settings:**
 - Name: nuget.org-proxy-v3
 - Format: nuget
 - Type: proxy
 - URL: http://localhost:8081/repository/nuget.org-proxy-v3/
 - Online: If checked, the repository accepts incoming requests
- NuGet:**
 - Metadata query cache age: 3600 seconds
- Proxy:**
 - Remote storage: https://api.nuget.org/v3/index.json

Figure: NuGet Proxy Repository Configuration for nuget.org V3

Once complete the V3 entry point for the proxy repository becomes: **<nexus-host>:<nexus-port>/repository/<repository-name>/index.json** (e.g., <http://localhost:8081/repository/nuget-proxy-v3/index.json>). This is the package source that should be passed to the nuget client in order to work with packages.

⁸¹³ <https://docs.microsoft.com/en-us/nuget/api/service-index>

⚠ Failure to use the correct NXRM URL will cause the repository to process requests using the V2 API. If the Remote Storage is pointing to a V3 Service Index this will provide no results for package requests and potentially cause build errors.

Example:

```
$ nuget list packageid:Newtonsoft.Json -Source http://localhost:8081/repository/nuget-proxy-v3/index.json
Newtonsoft.Json 12.0.3

$ nuget install Newtonsoft.Json -Source http://localhost:8081/repository/nuget-proxy-v3/index.json
Feeds used:
  http://localhost:8081/repository/nuget-proxy-v3/index.json

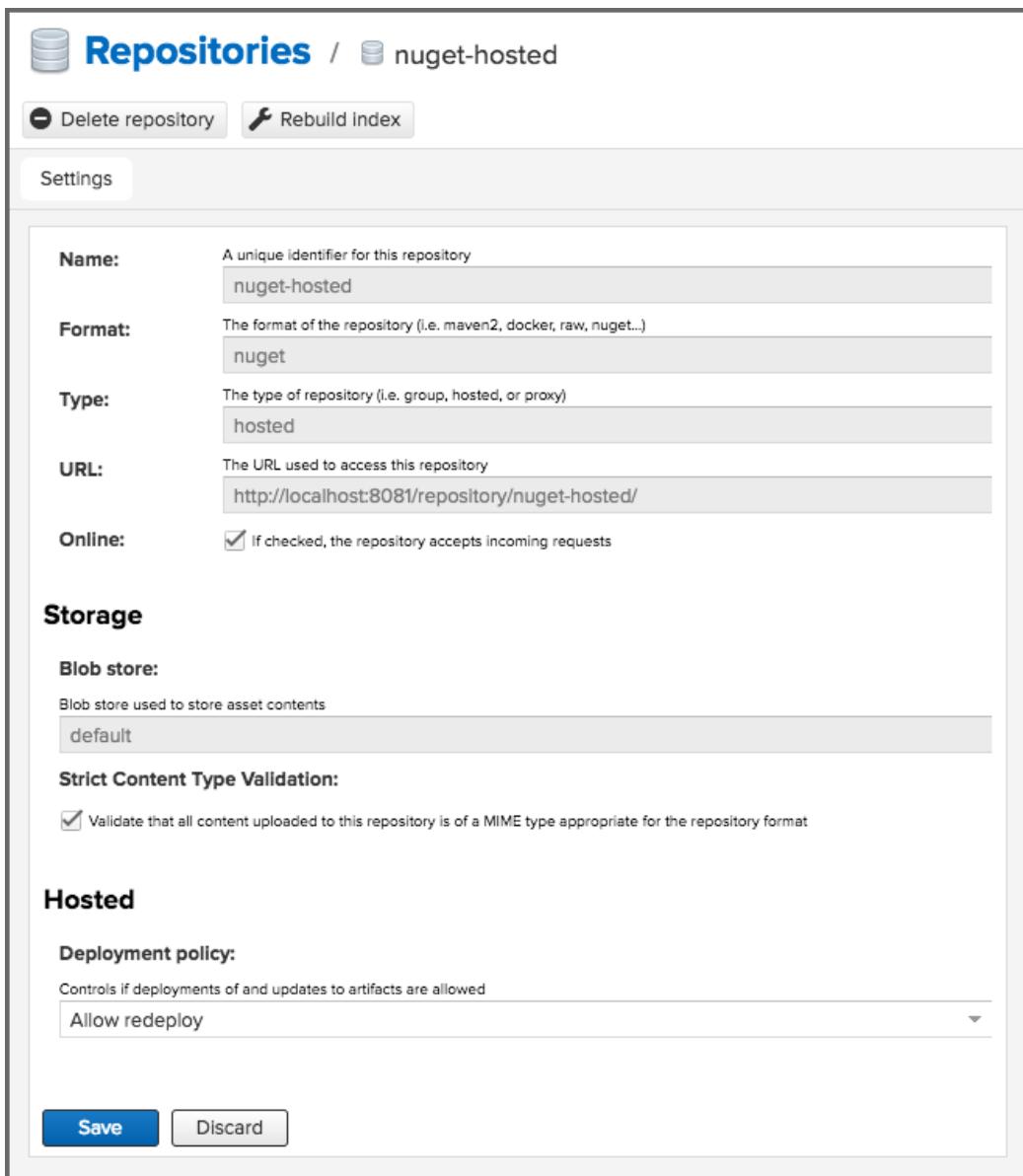
Installing package 'newtonsoft.json' to '/workspace/nuget'.
  CACHE http://localhost:8081/repository/nuget-proxy-v3/v3/registration/5/newtonsoft.json/index.json

Attempting to gather dependency information for package 'newtonsoft.json.12.0.3' with respect to project '/workspace/nuget', targeting 'Any,Version=v0.0'
Gathering dependency information took 19.78 ms
Attempting to resolve dependencies for package 'newtonsoft.json.12.0.3' with DependencyBehavior 'Lowest'
Resolving dependency information took 0 ms
Resolving actions to install package 'newtonsoft.json.12.0.3'
Resolved actions to install package 'newtonsoft.json.12.0.3'
Retrieving package 'Newtonsoft.Json 12.0.3' from 'http://localhost:8081/repository/nuget-proxy-v3/index.json'.
  GET http://localhost:8081/repository/nuget-proxy-v3/v3/content/0/newtonsoft.json/12.0.3/newtonsoft.json.12.0.3.nupkg
    OK http://localhost:8081/repository/nuget-proxy-v3/v3/content/0/newtonsoft.json/12.0.3/newtonsoft.json.12.0.3.nupkg 567ms
  Installing Newtonsoft.Json 12.0.3.
  Adding package 'Newtonsoft.Json.12.0.3' to folder '/workspace/nuget'
  Added package 'Newtonsoft.Json.12.0.3' to folder '/workspace/nuget'
  Successfully installed 'Newtonsoft.Json 12.0.3' to /workspace/nuget
  Executing nuget actions took 935.87 ms
```

NuGet Hosted Repositories

A hosted repository for NuGet can be used to upload your own packages as well as third-party packages. The repository manager includes a hosted NuGet repository named *nuget-hosted* by default.

To create another NuGet hosted repository, simply create a new *nuget (hosted)* repository. An example configuration from the default *nuget-hosted* repository is displayed in *Figure: "Example Configuration for a NuGet Hosted Repository"*.



The screenshot shows the 'Repositories' configuration page for a 'nuget-hosted' repository. The 'Settings' tab is selected. The configuration fields are as follows:

- Name:** nuget-hosted
- Format:** nuget
- Type:** hosted
- URL:** http://localhost:8081/repository/nuget-hosted/
- Online:**

Storage

Blob store: default

Strict Content Type Validation:

Hosted

Deployment policy: Allow redeploy

At the bottom are 'Save' and 'Discard' buttons.

Figure: Example Configuration for a NuGet Hosted Repository

The NuGet feed is immediately updated as packages are deployed or deleted from the host repository.

Grouping NuGet Repositories

A repository group is the recommended way to expose all your NuGet repositories from the repository manager to your users, without needing any further client side configuration. A repository group allows you to expose the aggregated content of multiple proxy and hosted repositories with one URL to your tools.

Nexus Repository Manager includes a *nuget-group* repository group. This default groups the *nuget.org-proxy* repository that proxies the NuGet Gallery and the *nuget-hosted* hosted repository. You can add

additional hosted and proxy repositories to it, once you have set them up, by following the concepts laid out in [Repository Group](#)(see page 240).

The URL of a repository group can be used in your client tool and will give you access to the packages in all repositories from the group with one URL. Any new packages added as well as any new repositories added to the group will automatically be available.

Accessing Packages in Repositories and Groups

You can access the repository group or individual repositories with the nuget tool on the command line by adding the URL to your nuget sources e.g.:

```
nuget sources add -name nuget-group -source http://localhost:8081/repository/nuget-group/
```

After this source was added, you can list the available packages with the command `nuget list`. It is worth noting, that by default `nuget.org` is listed as a source. If you don't disable or remove it, you will continue to fetch packages from your nuget commands regardless of changes made to your group.

Access to the packages is not restricted by default. If access restrictions are desired, you can configure security directly or via LDAP/Active Directory external role mappings combined with repository targets for fine grained control. Authentication from NuGet is then handled via NuGet API keys as documented in [Deploying Packages to NuGet Hosted Repositories](#)(see page 434).

Deploying Packages to NuGet Hosted Repositories

In order to authenticate a client against a NuGet repository, NuGet uses an API key for deployment requests. The API key acts as an alias for the user account, so the same API key is used for all NuGet repositories within the repository manager. This user-specific key is generated separately by a user and can be regenerated at any time. At regeneration, all previous keys generated for that user are invalidated.

Accessing your NuGet API Key

For usage with the repository manager, NuGet API keys are only needed when packages are going to be deployed. Users with the necessary `apikey-all` security privilege can access the *NuGet API Key* feature view via the User menu by pressing on their username in the main toolbar.

You can access your NuGet API key by pressing on the Access API Key button and providing password. The resulting user interface context is displayed in *Figure: “Accessing your NuGet API Key”*. It shows the key as well as the full command line to register the key for usage with nuget.

The Reset API Key button can be used to invalidate an existing API key and create a new one.

NuGet API Key

 Your NuGet API Key enables pushing packages using NuGet.exe. **Keep this key secret!**

Your NuGet API Key is:
3ec1ca19-0749-3654-a603-de3ed76299f6

You can register this key for a given repository with the following command:
`nuget setapikey 3ec1ca19-0749-3654-a603-de3ed76299f6 -source http://localhost`

This window will automatically close after one minute.

[Close](#)

Figure: Accessing your NuGet API Key

⚠ Usage of the API key requires the *NuGet API-Key Realm* to be activated. To do this, simply add the realm to the active realms in the *Realms* feature of the *Security* menu from the *Administration* menu. General details are available in [Realms](#)(see page 310).

Command line based Deployment to a NuGet Hosted Repository

The nuget command line tool allows you to deploy packages to a repository with the push command. The command requires you to use the NuGet API Key and the URL of the target hosted repository. For example, you could push to the default hosted repository using the URL `http://localhost:8081/repository/nuget-hosted`.

Using the delete command of nuget allows you to remove packages in a similar fashion. Further information about the command line tool is available in the [online help](#)⁸¹⁴.

Integration with Visual Studio

In order to access a NuGet repository or preferably all NuGet repositories exposed in a repository group, you provide the URL from the repository manager to configure Name and Source in the Visual Studio configuration for the Package Sources of the NuGet Package Manager as displayed in *Figure: “Package Source Configuration for the NuGet Package Manager in Visual Studio”*.

⁸¹⁴ <http://docs.nuget.org/docs/reference/command-line-reference>

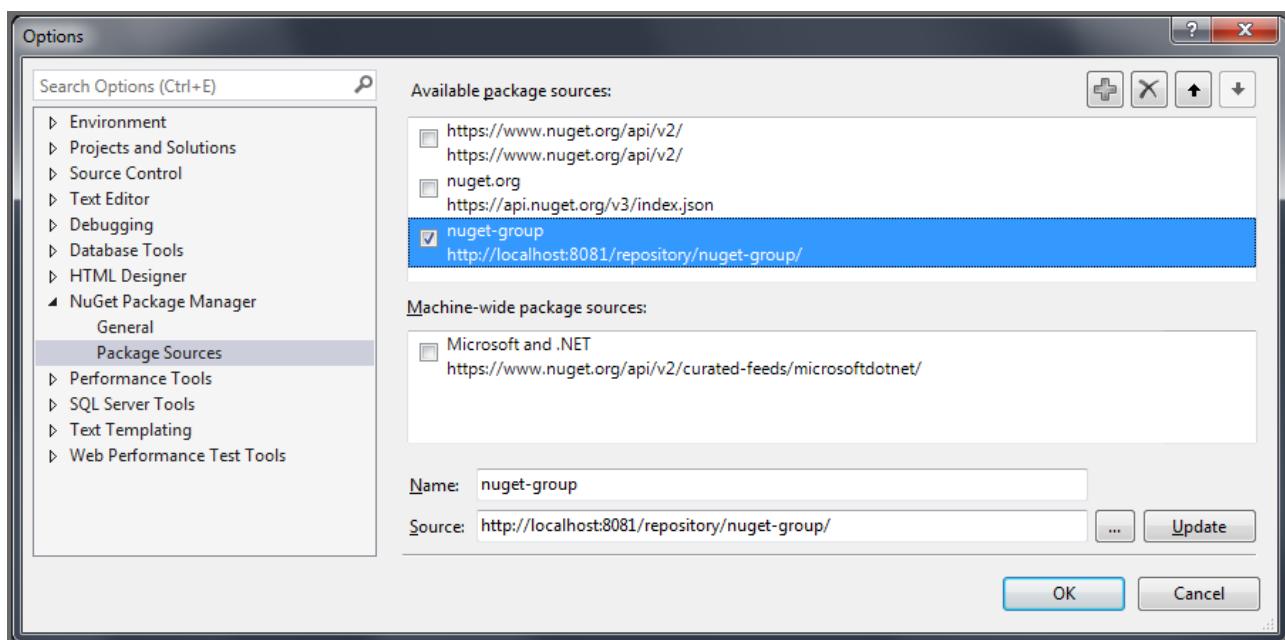


Figure: Package Source Configuration for the NuGet Package Manager in Visual Studio

With this configuration in place, all packages available in your NuGet repository will be available in the *NuGet Package Manager* in Visual Studio.

p2 Repositories

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

NEW IN RELEASE 3.21

Introduction

Fundamentally, p2 is a technology for provisioning and managing Eclipse- and Equinox-based applications. You can use p2 to install or manage any aspect of your application, from the physical plugins and native code, to the configuration of the installed software: file permissions, command line arguments, etc. Installation with p2 does not consist simply in adding or removing files in the file system, but more generally the sequence of events that must occur to lay down and configure a system that is ready to run.

 This format is disabled by default in NXRM Pro HA-C environments.

Supported features

- Repository types - Proxy
- Simple p2 plugin repositories
- Composite p2 plugin repositories
- Update sites
- Features, Plugins, Binary types of files
- Eclipse IDE version 4.2.2 or later (support for earlier versions is not guaranteed)

 Auto blocking unreachable remote repositories is disabled by default due to connection problems.

Proxying p2 Repositories

You can set up a p2 proxy repository to access a remote repository location, for example to proxy the Eclipse Foundation repository.

To proxy a p2 repository, you simply create a new *p2 (proxy)* as documented in [Repository Management\(see page 0\)](#) in details. Minimal configuration steps are:

- Define *Name* - e.g. p2-proxy
- Define *URL for Remote storage* - e.g. <http://download.eclipse.org/releases/2019-09>
- Select an appropriate *Blob store*

Browsing p2 Repositories and Searching Packages

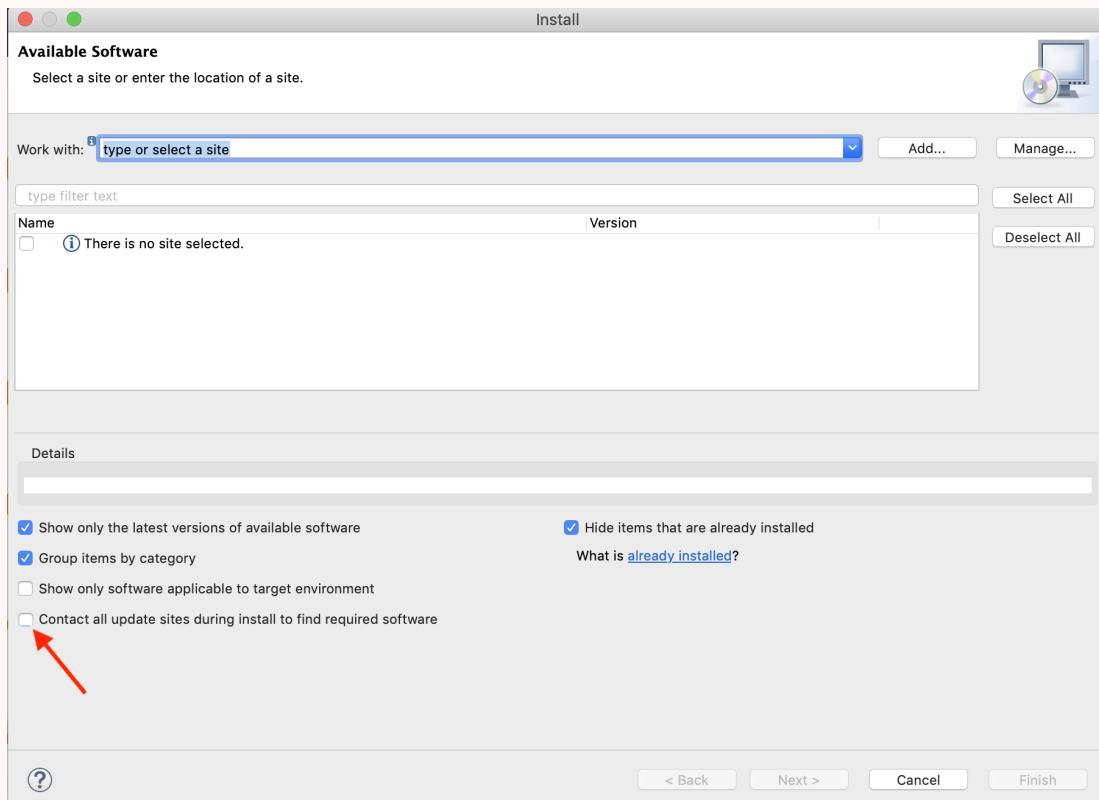
You can browse p2 repositories in the user interface inspecting the components and assets and their details, as described in [Browsing Repositories and Repository Groups\(see page 227\)](#).

Searching for p2 packages can be performed in the user interface, too. It finds all packages that are currently stored in the repository manager, as described in [Searching for Components\(see page 213\)](#).

Configuring the Eclipse IDE to use NXRM

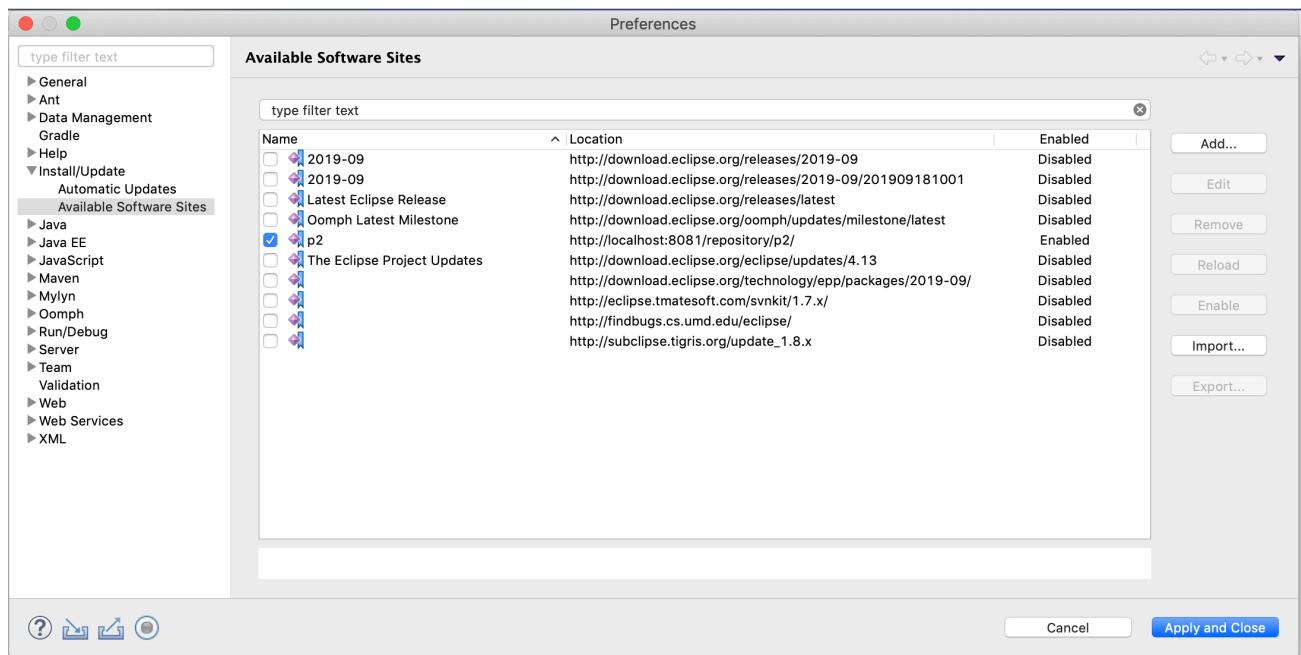
All p2 plugins are possible to install via Help → Install new software.

- ⚠ Please uncheck checkbox 'Contact all update sites during install to find required software' for installing directly from NXRM.



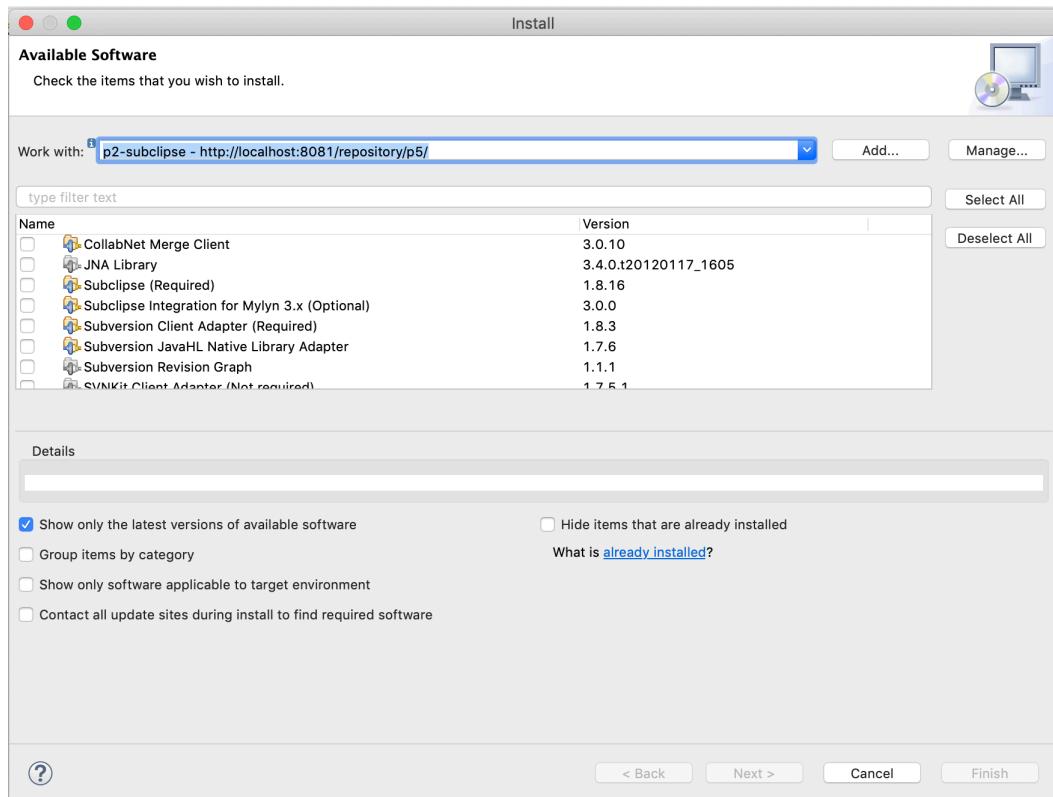
Click **Add** button to add plugin source and put in the URL field the NXRM proxy repository URL (e.g. `http://localhost:8081/repository/p2-proxy/`).

Click **Manage** button if you want to manage your sources. There are some Eclipse sources here by default, you can disable them if you need just NXRM sources (deleting these resources will not lead to the desired result, they will appear again after the application is restarted).

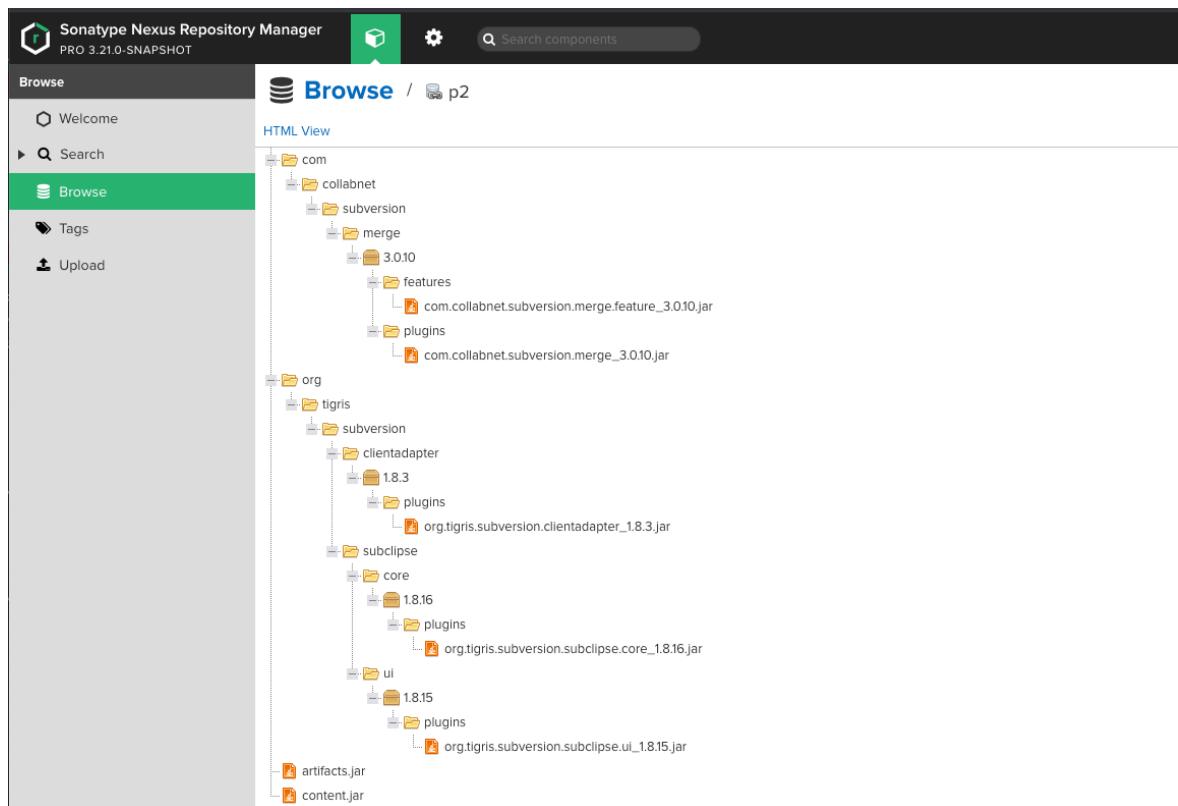


Plugins install

Select plugins



Follow the instructions. After installation NXRM contains proxied package, as shown:



PyPI Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

The Python Package Index, or PyPI, is a vast repository of open-source Python packages supplied by the worldwide community of Python developers. The official index is available at <https://pypi.org>, and the site itself is maintained by the [Python Software Foundation](#)⁸¹⁵.

Both Nexus Repository Manager Pro and Nexus Repository Manager OSS support proxying the Python Package Index. This allows the repository manager to take advantage of the packages in the official Python Package Index without incurring repeated downloads. This will reduce time and bandwidth usage for accessing Python packages.

⁸¹⁵ <https://www.python.org/psf/>

Also, you can publish your own packages to a private index as a hosted repository on the repository manager, then expose the remote and private repositories as a repository group, which is a repository that merges and exposes the contents of multiple repositories in one convenient URL.

- ⓘ If using pip with the repository manager, you should consider setting up your repository manager to use SSL as documented in [Configuring SSL\(see page 338\)](#). Otherwise, you will likely need to put --trusted-host additions at the end of many commands or further configure pip to trust your repository manager.

Proxying PyPI Repositories

You can set up a PyPI proxy repository to access a remote package index. To proxy a PyPI package index, you simply create a new *pypi (proxy)* recipe as documented in [Proxy Repository\(see page 238\)](#), in detail.

Minimal configuration steps are:

- Define *Name* - e.g. *pypi-proxy*
- Define URL for *Remote storage*. The official Python Package Index Remote Storage URL value to enter is `https://pypi.org/`. Using `https://pypi.python.org/` should also work as long as redirects are maintained.

The repository manager can access Python packages and tools from the remote index. The proxy repository for PyPI packages provides a cache of files available on the index making access to components from the Python Package Index more reliable. Users will be able to browse and search assets against the remote, as mentioned in [Browsing PyPI Repositories and Searching Packages\(see page 445\)](#).

Hosting PyPI Repositories

Creating a PyPI hosted repository allows you to upload packages in the repository manager. The hosted repository acts as an authoritative location for packages fetched from the Python index.

To host a PyPI package, create a new *pypi (hosted)* recipe as documented in [Hosted Repository\(see page 239\)](#), in detail.

Minimal configuration steps are:

- Define *Name* - e.g. *pypi-internal*
- Pick a *Blob store for Storage*

Grouping PyPI Repositories

A repository group is the recommended way to expose all your PyPI repositories from the repository manager to your users, with minimal additional client side configuration. A repository group allows you to expose the aggregated content of multiple proxy and hosted repositories as well as other repository groups with one URL in tool configuration. PyPI group repositories can be created with the `pypi (group)` recipe as documented in [Repository Group](#)(see page 240).

Minimal configuration steps are:

- Define *Name* - e.g. `pypi-all`
- Pick a *Blob store for Storage*
- Add PyPI repositories to the *Members* list in the desired order

Installing PyPI Client Tools

The latest versions of such Linux distributions as CentOS and Ubuntu come packaged with Python and `pip`⁸¹⁶, a tool for installing and managing Python packages from the index. For Mac OS X and Microsoft Windows, download and install a Python version compatible with the repository manager from <https://www.python.org/downloads/>. Download the pip installer from <https://pip.pypa.io/en/stable/installing/>.

 Nexus Repository Manager Pro and Nexus Repository Manager OSS support only specific versions of Python, pip, and setuptools. For Python, only the latest two releases 2 and 3 are supported. For pip and setuptools the latest two versions are supported. Versions not listed may work but are unlikely to be addressed if there are issues.

Configuring PyPI Client Tools

 Depending on your preference for either `setuptools`⁸¹⁷, `twine`⁸¹⁸, `distutils`⁸¹⁹ and pip, your proxy and hosted configuration will vary.

Once you have installed all necessary client tools from the Python Package Index, you can create and configure a `.pypirc` file to reference packages stored in the repository manager. Depending on your Python

⁸¹⁶ <https://pip.pypa.io/en/stable/>

⁸¹⁷ <https://pypi.python.org/pypi/setuptools>

⁸¹⁸ <https://pypi.python.org/pypi/twine>

⁸¹⁹ <https://docs.python.org/2.7/library/distutils.html>

configuration you can manage your repository groups with `pip.conf` or `setup.cfg` to have all commands, such as search and install, run against your project.

Download and install packages using `easy_install`

Easy Install lets you download, build, install, and update Python packages. Create a `setup.cfg` file which sets the `index-url` to the group, proxy or hosted repository from which you want to download packages.

In this example `index-url` is set for a proxy repository:

```
[easy_install]
index-url = http://localhost:8081/repository/pypi-proxy/simple
```

If you prefer to configure `easy_install` for hosted (`pypi-internal`) or group (`pypi-all`) adjust the file accordingly.

To install a package from the repository:

```
easy_install example-package
```

Upload to a hosted repository using `twine`

If you are authoring your own packages and want to distribute them to other users in your organization, you have to upload them to a hosted repository on the repository manager using a client tool such as `twine`.

The `.pypirc` holds your credentials for authentication when hosting a PyPI repository.

In the example `.pypirc` file below, specify the URL you want to deploy to the target hosted repository in the `repository` value. Add username and password values to access the repository manager. The `.pypirc` file contains `distutils`, a default server used by PyPI that provides upload commands that stores assets and authentication information.

```
[distutils]
index-servers =
pypi
[pypi]
repository: http://localhost:8081/repository/pypi-internal/
username: admin
password: admin123
```

 If you have multiple hosted repositories, you can add them to the `.pypirc` file, each with a different name, pointing to the corresponding repository URL.

After this is configured, you can upload packages to the hosted repository, as explained in [Uploading PyPI Packages](#)(see page 445).

Download, search and install packages using `pip`

To configure pip, create a `pip.conf`⁸²⁰ file on a Unix environment or a `pip.ini` file on Windows.

If you want pip to install or search Python within a group, configure the file to include the repository group URL.

```
[global]
index = http://localhost:8081/repository/pypi-all/pypi
index-url = http://localhost:8081/repository/pypi-all/simple
```

If you prefer to configure pip for proxy (e.g. pypi-proxy) or hosted (e.g. pypi-internal) adjust the file accordingly.

- ✓ To verify your configuration is correct run "pip config list -v". That will print the configuration being used, and the source configuration files it came from.

SSL Usage for PyPI Repositories

You can proxy Python packages over HTTPS to ensure a secure connection with a self-signed certificate. This works for proxy, hosted, and group repositories. To set up the repository manager to serve HTTPS follow the configuration steps in [Configuring SSL](#)(see page 338).

Also, you can set up pip to use the certificate to enable SSL and fetch packages securely. Additional configuration is necessary for the HTTPS client implementation to work. This assumes the repository manager has already been set up to use SSL, so verify your certificate works. Run the following command:

```
openssl verify <example-certificate>
```

When your certificate is proven to work, update your `pip.conf`. Here is an example configuration file for a repository group:

```
[global]
index = https://localhost:8443/repository/pypi-all/pypi
index-url = https://localhost:8443/repository/pypi-all/simple
cert = nexus.pem
```

⁸²⁰ https://pip.pypa.io/en/stable/user_guide/#config-file

Browsing PyPI Repositories and Searching Packages

You can browse PyPI repositories in the user interface inspecting the components and assets and their details, as described in [Browsing Repositories and Repository Groups](#)(see page 227).

Searching for PyPI packages can be performed in the user interface, as described in [Searching for Components](#)(see page 213). It finds all packages that are currently stored in the repository manager, either because they have been pushed to a hosted repository or they have been proxied from an upstream repository and cached in the repository manager.

From the command line you can search available PyPI packages defined in your configuration. This method is limited to pip (pip.conf).

To search, run:

```
pip search example-package
```

Uploading PyPI Packages

 The steps to upload a PyPI package will vary if your system is configured with setuptools or twine.

After you configure your .pypirc you can upload packages from the index to the repository manager.

In the example below, twine is invoked to tell your repository what server to use when uploading a package. The -r flag is used to find the NXRM server in your .pypirc.

```
twine upload -r pypi <filename>
```

R Repositories

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

NEW IN RELEASE 3.20

Introduction

R⁸²¹ is a language used for statistical analysis and machine learning. R and R Studio both allow you to install packages from repositories, allowing convenient access to a large amount of packages from the remote Comprehensive R Archive Network ([CRAN](#)⁸²²).

Nexus Repository Manager takes advantage of the packages in the official CRAN repository and other public repositories without incurring repeated downloads of packages.

You can publish your own packages to a private repository in a hosted R repository and then combine the remote and private repositories to R as a repository group, which is a repository that merges and exposes the contents of multiple repositories in one convenient URL. This allows you to reduce time and bandwidth usage for accessing R packages as well as more easily share your packages within your organization.

 This format is disabled by default in NXRM Pro HA-C environments.

Proxying R Repositories

You can set up an R proxy repository to access a remote repository location.

To proxy an R repository, create a new *r (proxy)* as shown in the documented example in [Repository Management](#)(see page 250) in detail. Minimal configuration steps for R proxy are:

- Define *Name*, e.g. r-proxy
- Define *URL for Remote storage*, e.g. <https://cran.r-project.org/>
- Select a *Blob store for Storage*

Hosting R Repositories

Creating an R hosted repository allows you to register packages in the repository manager. The hosted repository acts as an authoritative location for these components.

To add a hosted R repository, create a new repository with the recipe *r (hosted)* as shown in the documented example in [Repository Management](#)(see page 250). Minimal configuration steps for R hosted are:

- Define *Name*, e.g. r-hosted
- Select *Blob store for Storage*

⁸²¹ <https://www.r-project.org/>

⁸²² <https://cran.r-project.org/>

Deploying Packages to R Hosted Repositories

Upload with Nexus Repository Manager UI

Upload in the UI, as documented [here](#)⁸²³, to upload packages to R hosted repository.

R format requires to specify proper package path location **inside** a repository. You can do this by using the **package path** field in the Upload form. More details about the package path could be found on the official site (<https://cran.r-project.org>⁸²⁴) in the *Related Directories* section.

Several examples for package path field:

- For sources (*Package path*):
src/contrib
- For Windows binary (*Package path*): bin/windows/contrib/3.6
- For OS X binary (*Package path*): bin/macosx/el-capitan/contrib/3.6

Upload by HTTP POST

The following example uses the curl command to upload an example_1.0.0.tar.gz file to a hosted R repository (fill in <> with your respective data) :

```
curl -v --user '<username>:<password>' --upload-file example_1.0.0.tar.gz http://<host>:<port>/repository/<repository_name>/src/contrib/example_1.0.0.tar.gz
```

The package will now be contained within your hosted repository and consumers can install the package as described in [Configuring R Client](#)⁸²⁵.

R Repository Groups

A repository group is the recommended way to expose all your R repositories from the repository manager to your users, with minimal additional client-side configuration. A repository group allows you to expose the aggregated content of multiple proxy and hosted repositories as well as other repository groups with one URL in tool configuration. This is possible by creating a new repository with the *r (group)* recipe as shown in the documented example in [Repository Management](#)(see page 250). Minimal configuration steps for R group are:

⁸²³ <https://help.sonatype.com/repomanager3/user-interface/uploading-components>

⁸²⁴ <https://cran.r-project.org/web/packages/index.html>

⁸²⁵ <https://help.sonatype.com/display/NXRM3M/R+Repositories#RRepositories-ConfiguringRClient>

- Define *Name*, e.g. `r-group`
- Select *Blob store for Storage*
- Add R repositories to the *Members* list in the desired order

Configuring R Client

Once you have set up your repositories for R packages, you can adjust your R startup script to use your repository URLs. A suggested way to do so is to create a `.Rprofile` file and include a snippet similar to the following in it:

```
## Default repo
local({r <- getOption("repos")
      r["Nexus"] <- "http://<host>:<port>/repository/<repository_name>"}
      options(repos=r)
)}
```

This will set your default R repository as the group repository. For more information on adjusting R startup files, please visit the [mini guide on the r-bloggers](#)⁸²⁶.

Also, it is possible to install an R package directly from the R client console:

```
install.packages("example", repos="http://<host>:<port>/repository/<repository_name>",
type=<package_type>")
```

`<package_type>` is optional and accepts values of `source` and `binary`. If left off the console command, it defaults to `binary`.

If anonymous access to the repository manager is disabled, you have to specify the credentials for the accessing the repository manager as part of the URL like `http://<username>:<password>@<host>:<port>/repository/<repository_name>`

Downloaded packages are cached, do not have to be retrieved from the remote repositories again and can be inspected in the user interface.

Raw Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

⁸²⁶ <https://www.r-bloggers.com/fun-with-rprofile-and-customizing-r-startup>

Introduction

Nexus Repository Manager Pro and Nexus Repository Manager OSS include support for hosting, proxying and grouping static websites - the raw format. Hosted repositories with this format can be used to store and provide a Maven-generated website. Proxy repositories can subsequently proxy them in other servers. The raw format can also be used for other resources than HTML files exposed by straight HTTP-like browsable directory structures.

This section details the process of configuring raw repositories, configuring a simple Maven project to publish a Maven-generated project site and other use cases for raw repositories.

Creating a Hosted Raw Repository

To create a raw repository for hosting a static website, you simply create a new repository using the raw (hosted) recipe as documented in [Repository Management\(see page 234\)](#).

For the Maven site example in [Creating and Deploying a Maven Site\(see page 449\)](#), set the Name to `site` and change the Deployment policy to *Allow redeploy*.

After creating the new raw repository, it appears in the list of repositories with the name `site` provided earlier. The *URL* accessible in the list can be used for deployment and access usage.

- ⓘ Disable the *Strict Content Type Validation* if you encounter problems related to the content MIME-type.

Creating and Deploying a Maven Site

Creating a New Maven Project

In this section, you are be creating a minimal Maven project with a simple website that can be published to the hosted raw repository created in [Creating a Hosted Raw Repository\(see page 449\)](#).

The following steps can be used to create a new Maven project:

- Run the command `mvn archetype:generate` in a command line interface
- Confirm the first prompt using the default selection (number will vary)
- Confirm the default selection for the archetype version
- Set the `groupId` to `org.sonatype.books.nexus`
- Set the `artifactId` to `sample-site`
- Confirm the default version of `1.0-SNAPSHOT`

- Confirm the preset package of org.sonatype.books.nexus
- Confirm the properties configuration

After running the archetype:generate command, you will have a new project in a sample-site directory.

Configuring Maven for Site Deployment

To deploy a site to a raw repository in the repository manager, you need to configure the project's distributionManagement, add site deployment information, and then update your Maven settings to include the appropriate credentials.

Add the following section to sample-site/pom.xml before the dependencies element. This section tells Maven where to publish the Maven-generated project website.

Distribution Management for Site Deployment

```
<distributionManagement>
  <site>
    <id>nexus</id>
    <url>dav:http://localhost:8081/repository/site/</url>
  </site>
</distributionManagement>
```

The URL in the distribution management is not parameterized, which means that any redeployment overwrites old content and potentially leaves old stale files behind. To have a new deployment directory for each version, change the URL to a parameterized setup or change the whole URL between deployments.

If you combine this approach with a redirector or a static page that links to the different copies of your site, you can e.g., maintain separate sites hosting your javadoc and other documentation for different releases of your software.

The DAV protocol used by for deployment to the repository manager requires that you add the implementing library as a dependency of the Maven site plugin in your Maven project.

Dependency for the Maven Site Plugin for DAV Support

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-site-plugin</artifactId>
      <version>3.4</version>
      <dependencies>
        <dependency>
          <groupId>org.apache.maven.wagon</groupId>
          <artifactId>wagon-webdav-jackrabbit</artifactId>
          <version>2.8</version>
        </dependency>
      </dependencies>
    </plugin>
  </plugins>
</build>
```

Adding Credentials to Your Maven Settings

When the Maven site plugin deploys a site, it needs to supply the appropriate deployment credentials to the repository manager. To configure this, you need to add credentials to your Maven settings. Edit your `~/.m2/settings.xml` file and add the following server configuration to the `servers` element:

Configuring Deployment Credentials for Site Deployment

```
<settings>
  <servers>
    <!--your existing servers are here if any-->
    <server>
      <id>nexus</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
  </servers>
</settings>
```

⚠ Configuring Deployment Credentials for Site Deployment uses the default `admin` username and password. For real world usage you would use the username and password of a user with the privilege to write to the target repository.

Publishing a Maven Site

To publish the site to the hosted raw repository in the repository manager, run `mvn site-deploy` from the `sample-site` directory. The Maven site plugin will deploy this site to the repository using the credentials stored in your Maven settings:

Sample Maven Log from Deploying a Site

```
$ mvn site-deploy
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building sample-site 1.0-SNAPSHOT
...
[INFO] --- maven-site-plugin:3.4:site (default-site) @ sample-site ---
...
[INFO] Generating "About" report.
...
[INFO] --- maven-site-plugin:3.4:deploy (default-deploy) @ sample-site ---
http://localhost:8081/repository/site/ - Session: Opened
[INFO] Pushing /Users/manfred/training/sample-site/target/site
[INFO] &gt;&gt; to http://localhost:8081/repository/site/..
...
Transfer error: java.io.IOException: Unable to create collection: http://localhost:8081/repository/; status
code = 400
Uploading: ./project-summary.html to http://localhost:8081/repository/site/
##http://localhost:8081/repository/site./project-summary.html - Status code: 201
Transfer finished. 5078 bytes copied in 0.075 seconds
http://localhost:8081/repository/site/ - Session: Disconnecting
http://localhost:8081/repository/site/ - Session: Disconnected
...
[INFO] BUILD SUCCESS
...
```

Once the site has been published, you can load the site in a browser by going to `http://localhost:8081/repository/site/index.html`.



Figure: Maven-Created Sample Site Hosted in a Raw Repository

- ✓ A complete Maven project example can be found in the [documentation book examples](#)⁸²⁷.

Proxying and Grouping Raw Repositories

Beside the common use case using hosted raw repositories for site deployments, the repository manager supports proxying as well as grouping of raw repositories.

The creation follows the same process as documented in [Repository Management\(see page 0\)](#) using the *raw* (*proxy*) and the *raw* (*group*) recipes.

A raw proxy repository can be used to proxy any static website. This includes a Maven site hosted in a raw repository in another Nexus Repository Manager server or a plain static website hosted on another web server like Apache httpd. It can also be used to proxy directory structures exposed via a web server to distribute archives such as <https://nodejs.org/dist/>.

- ⚠ No content is modified when proxied. This means that e.g., any absolute URL used with HTML document remain absolute and therefore bypass the proxying mechanism.

Grouping raw repositories is possible and can e.g., be used to aggregate multiple site repositories. However keep in mind that the raw format does not contain any logic to resolve conflicts between the different repositories in the group. Any request to the group causes the repository manager to check the member repositories in order and return the first matching content.

⁸²⁷ <https://github.com/sonatype/nexus-book-examples/tree/nexus-3.x/maven/simple-project>

Uploading Files to Hosted Raw Repositories

Many other tools, besides using Maven, can be used to upload files to a hosted raw repository. A simple HTTP PUT can upload files. The following example uses the curl command and the default credentials of the admin user to upload a test.png file to a hosted raw repository with the name documentation.

An Example Upload Command Using curl

```
curl -v --user 'admin:admin123' --upload-file ./test.png http://localhost:8081/repository/documentation/test.png
```

After a completed upload the repository manager provides the file at the URL

<http://localhost:8081/repository/documentation/test.png>

. Using this approach in a script entire static websites or any other binary resources can be uploaded.

A complete static website consisting of numerous HTML, CSS, JS and image files or some other directory structure of multiple files and resources can be uploaded with a script that assembles and issues numerous HTTP PUT requests.

The [raw folder of the nexus-book-examples repository](#)⁸²⁸ contains the Groovy script rawPopulator.groovy as an example of such a script as well as a simplistic website of two HTML pages in the site directory. You can upload the directory to a raw repository with the name documentation to your repository manager at <http://repo.example.com:8081> with:

Example invocation of the rawPopulator script

```
$ groovy rawPopulator.groovy -d site -r documentation -u admin -p admin123 -h http://repo.example.com:8081
Staging 2 files for publishing
pushing site/index.html
POST response status: HTTP/1.1 201 Created
pushing site/success.html
POST response status: HTTP/1.1 201 Created
```

After the upload, you can access the site at <http://repo.example.com:8081/repository/documentation/> to load index.html and clicking on the link directs you to the success.html page.

RubyGems Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

⁸²⁸ <https://github.com/sonatype/nexus-book-examples/tree/nexus-3.x/raw>

Introduction

For developers using the Ruby programming language, the `gem` tool serves as their package management solution. In fact, since version 1.9 of Ruby, it has been included as part of the default Ruby library. Packages are called `gems` and, just like all package managers, this allows for ease of use when distributing programs or libraries.

Of course, package management really only goes as far as improving distribution. A great feat certainly, but to really find success, a development community needs to exist. At the heart of every development community, especially those like Ruby, where open source projects are one of the most critical elements, the community needs a place to host and share their projects.

Enter RubyGems via rubygems.org⁸²⁹ - the most popular and leading gem hosting service supporting the Ruby community. Here, a large variety of open source Ruby projects supply their gems for download to all users.

Ruby has been a successful platform for developers for a long time now. The popularity of Ruby and therefore the usage of gems and gem repositories means that lots of teams are downloading and exchanging lots of components on a regular basis. Obviously, this can (and does) become a crunch on resources, not to mention a pain to manage.

Luckily Nexus Repository Manager Pro and Nexus Repository Manager OSS support RubyGem repositories. A user can connect to the repository manager to download gems from RubyGems, create proxies to other repositories, and host their own or third-party gems. Any gem downloaded via the repository manager needs to be downloaded from the remote repository, like RubyGems, only once and is then available internally from the repository manager. Gems pushed to the repository manager automatically become available to everyone else in your organization. Using the repository manager as a proxy avoids the overhead of teams and individual developers having to repeatedly download components or share components in a haphazard and disorganized manner.

 Gem repository support is a feature of version NXRM 3.1 and higher.

The following features are included as part of the gem repository support:

- Proxy repository for connecting to remote gem repositories and caching gems on the repository manager to avoid duplicate downloads and wasted bandwidth and time
- Hosted repository for hosting gem packages and providing them to users
- Repository groups for merging multiple hosted and proxy gem repositories and easily exposing them as one URL to all users

⁸²⁹ <http://rubygems.org>

- ⓘ None of this functionality requires Ruby (or any extra tooling) to be installed on the operating system running the repository manager.

Proxying RubyGem Repositories

To reduce duplicate downloads and improve download speeds for your developers, continuous integration servers and other systems using `gem`, you should proxy the RubyGems repository and any other repositories you require.

To proxy an external gem repository, like RubyGems, you simply create a new repository using the recipe `rubygems (proxy)` as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Define URL for *Remote storage* e.g. `https://rubygems.org`
- Pick a *Blob store* for *Storage*

Further configuration details are available in [Managing Repositories and Repository Groups](#)(see page 240).

Private Hosted RubyGem Repositories

A private gem repository can be used as a target to push your own gems as well as third-party gems and subsequently provide them to your users. It is a good practice to create two separate hosted gem repositories for internal and third-party gems.

To create a hosted gem repository, create a new repository using the recipe `rubygems (hosted)` as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Select a *Blob store* for *Storage*

The gem repository information is immediately updated as gems are pushed to the repository or deleted from it.

Grouping RubyGem Repositories

A repository group is the recommended way to expose all your gem repositories to your users, without needing any further client side configuration after initial setup. A repository group allows you to expose the aggregated content of multiple proxy and hosted gem repositories with one URL to gem and other tools.

To create a gem group repository, create a new repository using the recipe *rubygems (group)* as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Select *Blob store for Storage*
- Add Gems repositories to the *Members* list in the desired order

A typical, useful example would be to group the proxy repository that proxies the RubyGems repository, a hosted gem repository with internal software gems, and another hosted gem repository with third-party gems.

Using the repository URL of the repository group as your gem repository URL in your client tool gives you access to the gems in all member repositories with one URL.

Any gem added to a hosted or proxy repository becomes immediately available to all users of the gem repository group. Adding a new proxy gem repository to the group makes all gems in that proxy immediately available to the users as well.

Using RubyGem Repositories

Once you have configured the repository manager with the gem repository group, you can add it to your configuration for the `gem` command line tool.

You can add the URL of a gem repository or group using the URL from the repository list with a command like:

```
$ gem sources --add http://localhost:8081/repository/rubygems-group/
```

In order to take full advantage of the repository manager and the proxying of gems, you should remove any other sources. By default <https://rubygems.org/> is configured in gem and this can be removed with:

```
$ gem sources --remove https://rubygems.org/
```

Subsequently you should clear the local cache with:

```
$ gem sources -c
```

To check a successful configuration you can run:

```
$ gem sources  
*** CURRENT SOURCES ***  
  
http://localhost:8081/repository/rubygems-group/
```

With this setup completed, any installation of new gems with `gem install <gem_name>` (e.g. `gem install rake`) will download from the repository manager.

If your repository manager requires authentication, you have to add the *Basic Auth* authentication details to the sources configuration:

```
$ gem sources --add http://myuser:mypassword@localhost:8081/repository/rubygems-group/
```

If you are using the popular [Bundler tool](#)⁸³⁰ for tracking and installing gems, you need to install it with `gem`:

```
$ gem install bundle  
Fetching: bundler-1.7.7.gem (100%)  
Successfully installed bundler-1.7.7  
Fetching: bundle-0.0.1.gem (100%)  
Successfully installed bundle-0.0.1  
Parsing documentation for bundle-0.0.1  
Installing ri documentation for bundle-0.0.1  
Parsing documentation for bundler-1.7.7  
Installing ri documentation for bundler-1.7.7  
Done installing documentation for bundle, bundler after 4 seconds  
2 gems installed
```

To use the repository manager with Bundler, you have to configure the gem repository group as a mirror:

```
$ bundle config mirror.http://rubygems.org  
http://localhost:8081/repository/rubygems-group/
```

You can confirm the configuration succeeded by checking the configuration:

```
$ bundle config  
Settings are listed in order of priority. The top value will be used.  
mirror.http://rubygems.org  
Set for the current user (/Users/manfred/.bundle/config): "http://localhost:8081/repository/rubygems-group"
```

With this configuration completed, you can create a `Gemfile` and run `bundle install` as usual and any downloads of gem files will be using the gem repository group configured as a mirror.

⁸³⁰ <http://bundler.io/>

Pushing Gems

At this point you have set up the various gem repositories on the repository manager (proxy, hosted and group), and are successfully using them for installing new gems on your systems. A next step can be to push gems to hosted gem repositories to provide them to other users. All this can be achieved on the command line with the features of the `nexus` gem.

The `nexus` gem is available at RubyGems and provides features to interact with Nexus Repository Manager Pro including pushing gems to a hosted gem repository including the necessary authentication.

You can install the `nexus` gem with:

```
$ gem install nexus
Fetching: nexus-1.2.1.gem (100%)
...
Successfully installed nexus-1.2.1
Parsing documentation for nexus-1.2.1
Installing ri documentation for nexus-1.2.1
Done installing
```

After successful installation you can push your gem to a desired repository. The initial invocation will request the URL for the gem repository and the credentials needed for deployment. Subsequent pushes will use the cached information.

```
$ gem nexus example-1.0.0.gem
Enter the URL of the rubygems repository on a Nexus server
URL: http://localhost:8081/repository/rubygems-hosted
The Nexus URL has been stored in ~/.gem/nexus
Enter your Nexus credentials
Username: admin
Password:
Your Nexus credentials has been stored in /Users/manfred/.gem/nexus
Uploading gem to Nexus...
Created
```

By default pushing an identical version to the repository, known as redeployment, is not allowed in a hosted gem repository. If desired this configuration can be changed, although we suggest to change the version for each new deployment instead.

The `nexus` gem provides a number of additional features and parameters. You can access the documentation with:

```
$ gem help nexus
```

E.g. you can access a list of all configured repositories with:

```
$ gem nexus --all-repos
DEFAULT: http://localhost:8081/repository/rubygems-hosted
```

Yum Repositories

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

[Yum or "Yellowdog Updater, Modified"](#)⁸³¹ is a command line package management utility for Linux distributions using the RPM package manager. It allows for many commonly used Linux packages to be easily installed on to distributions such as RedHat, CentOS and Fedora.

Nexus Repository Manager OSS and Nexus Repository Manager Pro support the Yum repository format for proxy repositories. This allows the repository manager to take advantage of the packages in public Yum repositories without incurring repeated downloads of packages. This will also allow you to perform offline installs, for example allowing you to install CentOS without a connection to the internet.

 Important

Upgrading Yum repositories from version 2 to version 3 is currently not supported.

Proxying Yum Repositories

You can set up a Yum proxy repository to access a remote repository location.

To proxy a Yum repository, you simply create a new *yum (proxy)* as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define Name e.g. yum-proxy
- Define URL for Remote storage, e.g. <http://mirror.centos.org/centos/>
- Pick a Blob store for Storage

⁸³¹ <http://yum.baseurl.org/>

⚠ We do not create a default Yum proxy repository as there are many. Determine which repositories are appropriate for your environment.

Hosting Yum Repositories

A hosted repository for Yum can be used to upload your own RPMs as well as third-party RPMs. To host a Yum RPM, create a new *yum (hosted)* repository as documented in [Repository Management](#)(see page 234).

When creating a Yum Hosted repository, you'll need to pick a *Repodata Depth*. This sets the level that the *repodata* metadata folder will be created initially as well as the expected folder depth that the rpms can be at (or deeper than) to match. Rpm's with less depth will be rejected by Nexus Repository Manager.

For example, if your package is created at `/games/lol/ashe.rpm` then the *Repodata Depth* would be 2 and when pushing the rpm in the metadata would be created at that level. However, if you also had `/games/poker.rpm` then you'd want to have *Repodata Depth* equal to 1, which would account for both the `poker.rpm` and the `ashe.rpm` in this example. If you had *Repodata Depth* as 2 and tried to push `/games/poker.rpm` it'd be rejected. In either case, pushing `/games-wow/horde/thrall.rpm` would be fine, it having a depth more than both 1 and 2.

It is possible to configure a *Layout Policy* which defaults to *Strict*. When this is set to *Strict*, only yum-specific files (`rpm`, `comps.xml`) can be uploaded. When this is set to *Permissive*, any file of any type can be uploaded.

⚠ A *Layout Policy* of *Permissive* is required to use the Maven deploy plugin with a Yum Hosted repository.

Minimal configuration steps for creating a Yum Hosted repository are:

- Define *Name* e.g. `yum-hosted`
- Select a value for *Repodata Depth*
- Pick a *Blob store* for *Storage*

⚠ *Repodata Depth* is an editable field after repository creation, however, it is not a recommended practice. Adjusting it to a value that's deeper than you have existing data will cause issues with your repository which would need to be resolved by a *Rebuild Yum Metadata* task run, after the value is corrected.

Grouping Yum Repositories

A repository group for Yum allows you to expose the aggregated content of multiple proxy and hosted yum repositories with one URL to your client tools and is recommended to minimize configuration.

To create a yum group repository, create a new repository using the recipe *yum (group)* as documented in [Repository Management](#)(see page 234).

Minimal configuration steps are:

- Define *Name*
- Select *Blob store for Storage*
- Add Yum repositories to the *Members* list in the desired order

A typical, useful example would be to group the proxy repository that proxies an external Yum repository (for example CentOS), a hosted Yum repository for internal rpms, and another hosted Yum repository for third-party rpms.

Using the *repository URL* of the repository group as your Yum *baseURL* in your client tool gives you access to the rpms in all member repositories with one URL.

Any rpm added to a hosted or proxy repository becomes immediately available to all users of the Yum repository group.

 Unlike Yum Hosted, metadata for a Yum Group is not generated until a request or search is made against the group.

Yum group will only merge content of members that are using the same endpoint as other members. As an example, to be able to use a Yum group endpoint that contains a hosted repository and a proxied CentOS (configured endpoint `http://mirror.centos.org/centos/$version/os/$arch`), your hosted structure should follow the same endpoint convention, i.e. using a repodepth of 3 and pushing your rpms to `http://<ip-address>/repository/yum-hosted/7/os/x86_64/example.rpm` for centos7 OS and x86_64 architecture respectively.

Deviating from a common structure will not merge metadata; taking the example above if the rpms were pushed to the hosted repository as `http://localhost:8081/repository/yum-hosted/extras/x86_64/example.rpm` the metadata will be available for use but it will not be merged. See specific details on how to configure your client [in below subsection](#)(see page 463).

Deploying Packages to Yum Hosted Repositories

The Yum client does not come with a method for uploading RPMs however many other tools can be used to upload files to a hosted Yum repository using a simple HTTP PUT. The following example uses the curl command and the default credentials of the admin user to upload a `test.rpm` file to a hosted Yum repository with the name `test.rpm`:

```
curl -v --user 'admin:admin123' --upload-file ./test.rpm http://localhost:8081/repository/yum-hosted/test.rpm
```

When an RPM is uploaded, the Yum metadata will be generated after the configured value (default 60 seconds).

The task *Repair - Rebuild Yum repository metadata (repodata)* can also be configured to create the metadata if the standard generation fails. This is not typically needed, thus is regarded as a Repair task. Further information about tasks can be found in [Configuring and Executing Tasks](#)(see page 269).

Comps.xml (package grouping)

Yum package groups are supported by uploading a comps.xml file or a comps.xml.gz file (the exact filename must be used for it to be detected by the repository manager and included in the repomd.xml file):

```
curl -v --user 'admin:admin123' --upload-file ./  
b686d3a0f337323e656d9387b9a76ce6808b26255fc3a138b1a87d3b1cb95ed5-comps.xml http://localhost:8081/  
repository/yum-hosted/repodata/comps.xml  
curl -v --user 'admin:admin123' --upload-file ./  
b686d3a0f337323e656d9387b9a76ce6808b26255fc3a138b1a87d3b1cb95ed5-comps.xml.gz http://localhost:8081/  
repository/yum-hosted/repodata/comps.xml.gz
```

Installing Yum

Yum should come pre-installed with RedHat, CentOS, Fedora and a long list of Linux flavors. If your system does not have Yum preinstalled, you may have larger problems that cannot be solved in these docs.

⚠ Fedora users are encouraged to use <http://dnf.baseurl.org/> as of Fedora version 20. DNF is currently backwards compatible and should work with Nexus Repository Manager 3, but is not explicitly supported.

Configuring Yum Client

Create a nexus.repo file in /etc/yum.repos.d/ that looks similar to the following:

```
nexus.repo  
  
[nexusrepo]  
name=Nexus Repository  
baseurl=http://<serveraddress:port>/repository/yum-proxy/$releasever/os/$basearch/  
enabled=1  
gpgcheck=1  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7  
priority=1
```

⚠ If you have set gpgcheck to enabled, you'll want to provide the location of the gpgkey, replacing the value we've shown in the example above.

Browsing Yum Repositories and Searching Packages

You can browse Yum repositories in the user interface inspecting the components and assets and their details, as described in [Browsing Repositories and Repository Groups](#)(see page 227).

Searching for Yum packages can be performed in the user interface, too. It finds all packages that are currently stored in the repository manager, as described in [Searching for Components](#)(see page 213).

REST and Integration API

ⓘ Available in Nexus Repository OSS and Nexus Repository Pro

REST API

Nexus Repository Manager leverages a simple UI interface to generate the comprehensive REST documentation from the available endpoints. The REST API can be used to integrate the repository manager with external systems. The interface is fully interactive, where parameters can be filled out and REST calls made directly through the UI to see the results in the browser.

As of NXRM 3.6.1, this interface is available under the *API* item via the *System* sub menu of the *Administration* menu.

For older versions of NXRM (3.3.0 through 3.6.0), use the following:

```
<nexus_url>/swagger-ui/
```

A comprehensive listing of REST API endpoints and functionality is documented through the UI. The following highlights noteworthy functionality:

- Search - component search functionality
- Assets - direct access the assets/binaries and associated metadata in a repository
- Components - direct access to language native logical grouping of files in a repository, e.g. maven2 groupId, artifactId, version
- Script - for provisioning and other complex tasks

- Tasks - available administrative tasks for the repository manager

Topics in this section:

- [Assets API](#)(see page 466)
- [Blob Store API](#)(see page 468)
- [Components API](#)(see page 469)
- [Email API](#)(see page 477)
- [IQ Server API](#)(see page 477)
- [Licensing API](#)(see page 478)
- [Lifecycle API](#)(see page 478)
- [Maintenance API](#)(see page 480)
- [Nodes API](#)(see page 484)
- [Read-Only API](#)(see page 485)
- [Repositories API](#)(see page 488)
- [Search API](#)(see page 489)
- [Security Management API](#)(see page 496)
- [Script API](#)(see page 497)
- [Status API](#)(see page 509)
- [Support API](#)(see page 510)
- [Tasks API](#)(see page 511)
- [Pagination](#)(see page 514)

Scripting API

 **Disabled Groovy Scripting By Default**

NEW IN RELEASE 3.21.2

In order to make NXRM more secure, we have disabled Groovy scripting engine by default. This affects Groovy scripts as used through the REST API and through scheduled tasks.

For more information (including how to re-enable Groovy scripting), see [NEXUS-23205](#)⁸³².

⁸³² <https://issues.sonatype.org/browse/NEXUS-23205>

Nexus Repository Manager also provides a scripting API. Scripts can be written to perform custom tasks that can't be handled directly through the UI or the REST API. NXRM scripts are written in the Groovy programming language. More information about Groovy can be found at <http://groovy-lang.org/>.

There are several methods to run scripts, including:

- To run a script from the Nexus Repository UI, you must create a scheduled task. Navigate to *Tasks* from the *Administration* menu, and create a task of type *Admin - Execute script*. Enter the script source into the *Source* form field. Set the task frequency to *Manual* (or another setting to execute on a schedule) and save. The script can then be executed by selecting it from the list of tasks and pressing the *Run* button.
- A script may be uploaded and run through the REST [Script API](#)(see page 497)

Nexus provides the following APIs to scripts:

- Core API: allows a script to set global configuration options
- Security API: allows a script to add users, roles, and privileges
- Blob Store API: allows a script to create new blob stores
- Repository API: provides a simple, format-specific interface for creating repositories

Sample scripts and javadoc API documentation can be found at <https://github.com/sonatype-nexus-community/nexus-scripting-examples>.

For recommendations on configuring a stable scripting environment and tips from Sonatype, see [this knowledge base article](#)⁸³³.

Assets API

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

This set of endpoints is for interacting with assets directly.

⁸³³ <https://support.sonatype.com/hc/en-us/articles/115015812727-Nexus-3-Groovy-Script-development-environment-setup>

Endpoints

List Assets

```
GET /service/rest/v1/assets
```

This endpoint allows us to iterate through a listing of assets contained in a given repository.

Let's get a listing of the assets in the maven-central repository:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/assets?repository=maven-central'
```

This produces a response that is the first pages of many assets in that repository:

Example Response

```
{
  "items" : [ {
    "downloadUrl" : "http://localhost:8081/repository/maven-central/asm/asm/3.1/asm-3.1-sources.jar",
    "path" : "asm/asm/3.1/asm-3.1-sources.jar",
    "id" : "bWF2ZW4tY2VudHJhbD0zzjVjYWUwMTc2MDIzM2I2MTRkZWJlOTE0GFjNjMwOQ",
    "repository" : "maven-central",
    "format" : "maven2",
    "checksum" : {
      "sha1" : "2eaa4de56203f433f287a6df5885ef9ad3c5bcae",
      "md5" : "0c0e3e83d9ce7907334c5eb519dab10c"
    }
  },
  ...
  ],
  "continuationToken" : "3f5cae01760233b6506547dc7be10e0b"
}
```

This endpoint uses a [pagination⁸³⁴](#) strategy that can be used to iterate through all the assets if desired.

Note that the ordering of assets is consistent across multiple queries and is not alphabetical.

Get Asset

```
GET /service/rest/v1/assets/{id}
```

⁸³⁴ <https://help.sonatype.com/display/NXRM3M/Pagination>

This endpoint allows us to get the details of an individual asset.

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/assets/bWF2ZW4tY2VudHJhbDozzjVjYWUwMTc2MDIzM2I2MjRiOTEwMmMmNiYmU4YQ'
```

Example Response

```
{  
  "downloadUrl" : "http://localhost:8081/repository/maven-central/org/sonatype/nexus/buildsupport/nexus-buildsupport-metrics/2.9.1-02/nexus-buildsupport-metrics-2.9.1-02.pom",  
  "path" : "org/sonatype/nexus/buildsupport/nexus-buildsupport-metrics/2.9.1-02/nexus-buildsupport-metrics-2.9.1-02.pom",  
  "id" : "bWF2ZW4tY2VudHJhbDozzjVjYWUwMTc2MDIzM2I2MjRiOTEwMmMmNiYmU4YQ",  
  "repository" : "maven-central",  
  "format" : "maven2",  
  "checksum" : {  
    "sha1" : "a3bf672b3ea844575acba3b84790e76ed86a7c66",  
    "md5" : "49e439c814c3098450dc4bbe952463f"  
  }  
}
```

Delete Asset

```
DELETE /service/rest/v1/assets/{id}
```

This endpoint can be used to delete an individual asset.

```
curl -u admin:admin123 -X DELETE 'http://localhost:8081/service/rest/v1/assets/bWF2ZW4tY2VudHJhbDozzjVjYWUwMTc2MDIzM2I2MjRiOTEwMmMmNiYmU4YQ'
```

A successful deletion will return no content, for example:

Example Response

```
HTTP/1.1 204 No Content  
Date: Fri, 19 Jan 2018 20:41:47 GMT  
...
```

Blob Store API

Introduction

The [Blob Store](#)(see page 281) exposes HTTP endpoints for creating, updating, deleting, retrieving and listing blob stores.

Please see the *API* section (found in the *Administration* section under *System → API*) of your Nexus Repository Manager instance for detailed information about the *Blob Store API* including sample requests and responses.

Endpoints

NEW IN 3.19 The blobstores REST API endpoints can be used to accomplish the following:

- List blob stores
- Delete blob store
- Create blob store
- Get blob store configuration
- Update blob store
- Get blob store [quota](#)(see page 237)

Components API

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

Introduction

This set of endpoints is for interacting with components directly.

Endpoints

List Components

```
GET /service/rest/v1/components
```

This endpoint allows us to iterate through a listing of components contained in a given repository.

Let's get a listing of the components in the
maven-central
repository:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/components?repository=maven-central'
```

Typically this will produce a response that is the first page of many components in that repository, as such:

Example Response

```
{  
  "items" : [ {  
    "id" : "bWF2ZW4tY2VudHJhbDo40DQ5MWNkMWQxODVkJDEzNjMyNjhMjIwZTQ1ZDdkZQ",  
    "repository" : "maven-central",  
    "format" : "maven2",  
    "group" : "com.google.guava",  
    "name" : "guava",  
    "version" : "21.0",  
    "assets" : [ {  
      "downloadUrl" : "http://localhost:8081/repository/maven-central/com/google/guava/guava/21.0/  
guava-21.0.jar",  
      "path" : "com/google/guava/21.0/guava-21.0.jar",  
      "id" : "bWF2ZW4tY2VudHJhbDozZjVjYWUwMTc2MDIzM2I2MzA40ThiZjZmZTFkOTE2NA",  
      "repository" : "maven-central",  
      "format" : "maven2",  
      "checksum" : {  
        "sha1" : "3a3d111be1be1b745edfa7d91678a12d7ed38709",  
        "md5" : "ddc91fd850fa6177c91aab5d4e4d1fa6"  
      }  
    }, {  
      "downloadUrl" : "http://localhost:8081/repository/maven-central/com/google/guava/guava/21.0/  
guava-21.0.jar.sha1",  
      "path" : "com/google/guava/21.0/guava-21.0.jar.sha1",  
      "id" : "bWF2ZW4tY2VudHJhbDpm0Dk4YjM5MDNjYjk5YzU5MDc0MDFlYzRjNjVlNju50Q",  
      "repository" : "maven-central",  
      "format" : "maven2",  
      "checksum" : {  
        "sha1" : "a1ff60cb911e1f64801c03d03702044d10c9bdd3",  
        "md5" : "e34b8695ede1677ba262411d757ea980"  
      }  
    }, {  
      "downloadUrl" : "http://localhost:8081/repository/maven-central/com/google/guava/guava/21.0/  
guava-21.0.pom",  
      "path" : "com/google/guava/21.0/guava-21.0.pom",  
      "id" : "bWF2ZW4tY2VudHJhbDpkMDY0ODA0YThlZDVhZDZl0WJjNDgZ0GE1MzM20GzlZg",  
      "repository" : "maven-central",  
      "format" : "maven2",  
      "checksum" : {  
        "sha1" : "fe4fa08a8c0897f9896c7e278fb397ede4a2feed",  
        "md5" : "5c10f97af2ce9db54fa6c2ea6997a8d7"  
      }  
    }, {  
      "downloadUrl" : "http://localhost:8081/repository/maven-central/com/google/guava/guava/21.0/  
guava-21.0.pom.sha1",  
      "path" : "com/google/guava/21.0/guava-21.0.pom.sha1",  
      "id" : "bWF2ZW4tY2VudHJhbDplMDE40GVkMDcy0GZhNjhZDA3NDdkNjlhZDNmZjI5Nw",  
      "repository" : "maven-central",  
      "format" : "maven2",  
      "checksum" : {  
        "sha1" : "992b43ab7b3a061be47767e910cab58180325abc",  
        "md5" : "33aed29aa0bb4e03ea7854066a5b4738"  
      }  
  }]
```

```
    } ],
},
...
],
"continuationToken" : "88491cd1d185dd136f143f20c4e7d50c"
}
```

This endpoint uses a [pagination](#)(see page 514) strategy that can be used to iterate through all the components if desired.

Note that the ordering of the components is consistent across multiple queries and it is not alphabetical.

Get Component

```
GET /service/rest/v1/components/{id}
```

This endpoint allows us to get details about an individual component.

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/components/bWF2ZW4tY2VudHJhbDo40DQ5MWNkMWQxODVkJDEzNjYwYmUwMjE1MjI2NGUwZQ'
```

Example Response

```
{  
  "id" : "bWF2ZW4tY2VudHJhbDo40DQ5MWNkMWQxODVkJDEzNjYwYmUwMjE1MjI2NGUwZQ",  
  "repository" : "maven-central",  
  "format" : "maven2",  
  "group" : "org.apache.httpcomponents",  
  "name" : "httpcomponents-client",  
  "version" : "4.3.5",  
  "assets" : [ {  
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/apache/httpcomponents/httpcomponents-client/4.3.5/httpcomponents-client-4.3.5.pom",  
    "path" : "org/apache/httpcomponents/httpcomponents-client/4.3.5/httpcomponents-client-4.3.5.pom",  
    "id" : "bWF2ZW4tY2VudHJhbDozZjVjYWUwMTc2MDIzM2I2YTFhOGUxOGQxZmFk0GM3Mw",  
    "repository" : "maven-central",  
    "format" : "maven2",  
    "checksum" : {  
      "sha1" : "95d80a44673358a5dcbcc2f510770b9f93fe5eba",  
      "md5" : "f4769c4e60799ede664414c26c6c5c9d"  
    }  
  }, {  
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/apache/httpcomponents/httpcomponents-client/4.3.5/httpcomponents-client-4.3.5.pom.sha1",  
    "path" : "org/apache/httpcomponents/httpcomponents-client/4.3.5/httpcomponents-client-4.3.5.pom.sha1",  
    "id" : "bWF2ZW4tY2VudHJhbDpmODk4YjM5MDNjYjk5YzU5ZDU3YjFlYjE0MzM1ZTcwMQ",  
    "repository" : "maven-central",  
    "format" : "maven2",  
    "checksum" : {  
      "sha1" : "6b98f5cef5d7102f8f45215bdcf48dc843d060af",  
      "md5" : "f3b3ac640853fc887621d13029a1747"  
    }  
  } ]  
}
```

Upload Component

```
POST /service/rest/v1/components
```

This endpoint allows us to upload a component to a specified repository. Some formats allow to include multiple assets in a single component (please refer to [Uploading Components](#) (see page 230) and more information below).

```
curl -v -u admin:admin123 -X POST 'http://localhost:8081/service/rest/v1/components?repository=maven-releases' -F maven2.groupId=com.google.guava -F maven2.artifactId=guava -F maven2.version=24.0-jre -F maven2.asset1=@guava-24.0-jre.jar -F maven2.asset1.extension=jar -F maven2.asset2=@guava-24.0-jre-sources.jar -F maven2.asset2.classifier=sources -F maven2.asset2.extension=jar
```

A successful upload will return no content, as follows:

```
HTTP/1.1 204 No Content
Date: Fri, 19 Jan 2018 20:26:13 GMT
...
```

The parameters expected by the endpoint depend on the format of the repository where we are going to upload our component.

Maven

Maven format allows multiple assets to be uploaded as part of a single component. To upload multiple assets just follow the information from a table describing the given format and replace

assetN

with multiple instances of it (e.g. asset1, asset2, etc.):

Field name	Field type	Required?	Description
maven2.groupId	String	Yes, unless a POM asset is included in the upload	Group ID of the component
maven2.artifactId	String	Yes, unless a POM asset is included in the upload	Artifact ID of the component
maven2.version	String	Yes, unless a POM asset is included in the upload	Version of the component
maven2.generate-pom	Boolean	No	Whether the Nexus Repository Manager should generate a POM file based on above component coordinates provided
maven2.packaging	String	No	Define component packaging (e.g. jar, ear)
maven2.assetN	File	Yes, at least one	Binary asset
maven2.assetN.extension	String	Yes	Extension of the corresponding assetN asset
maven2.assetN.classifier	String	No	Classifier of the corresponding assetN asset

Examples

Uploading a jar and Automatically Creating a pom File

For this example we have a single jar file from a third-party vendor and no project POM file from that third party. We have decided our own Maven 2 coordinates and will ask Nexus to automatically generate and store a POM file for us.

Example Maven jar Component Upload With Automatically Created pom File Using curl

```
curl -v -u admin:admin123 -F "maven2.generate-pom=true" -F "maven2.groupId=com.example" -F "maven2.artifactId=commercial-product" -F "maven2.packaging=jar" -F "version=1.0.0" -F "maven2.asset1=@/absolute/path/to/the/local/file/product.jar;type=application/java-archive" -F "maven2.asset1.extension=jar" "http://localhost:8081/service/rest/v1/components?repository=maven-third-party"
```

- **maven2.generate-pom=true** tells Nexus to generate a minimal POM file for us based on the groupId, artifactId, version and packaging we provide
- replace "admin:admin123" with your own valid username:password
- replace "/absolute/path/to/the/local/file/product.jar" with the local path to the file you are uploading, however leave the leading @ so that curl understands how to find this file
- ;**type=application/java-archive** is optional
- the repository **maven-third-party** must already exist or there will be an error
- -v makes the command verbose to help you verify a successful response is returned. Otherwise you wouldn't be able to easily tell if it worked or returned an HTTP error code

Upload a POM and associated JAR File

For this example we have a single jar file and we have already created a minimal project pom.xml which includes groupId, artifactId, version elements defined inside of it. Nexus will accept both the POM and jar file in a single upload operation, and will parse the Maven 2 coordinates from the uploaded POM file and install the artifacts to the correct locations inside the maven-releases repository. Notice how the command need not specify the Maven coordinates.

Example pom and jar Component Upload Using curl

```
curl -v -u admin:admin123 -F "maven2.generate-pom=false" -F "maven2.asset1=@/absolute/path/to/the/local/file/pom.xml" -F "maven2.asset1.extension=pom" -F "maven2.asset2=@/absolute/path/to/the/local/file/product-1.0.0.jar;type=application/java-archive" -F "maven2.asset2.extension=jar" "http://localhost:8081/service/rest/v1/components?repository=maven-releases"
```

Raw

Raw supports uploading multiple assets within a request

Field name	Field type	Required?	Description
raw.directory	String	Yes	Destination for upload files (e.g. /path/to/files)
raw.assetN	File	Yes, at least one	Binary asset

Field name	Field type	Required?	Description
raw.assetN.filename	String	Yes	Filename to be used for the corresponding assetN asset

PyPI

Field name	Field type	Required?	Description
pypi.asset	File	Yes	Binary asset

RubyGems

Field name	Field type	Required?	Description
rubygems.asset	File	Yes	Binary asset

NuGet

Field name	Field type	Required?	Description
nuget.asset	File	Yes	Binary asset

NPM

Field name	Field type	Required?	Description
npm.asset	File	Yes	Binary asset

Delete Component

```
DELETE /service/rest/v1/components/{id}
```

This endpoint can be used to delete an individual component along with the associated assets.

```
curl -u admin:admin123 -X DELETE 'http://localhost:8081/service/rest/v1/components/bWFZtY2VudHJhbDo40DQ5MWNkMWQxODVkJDEzNjYwYmUwMjE1MjI2NGUwZQ'
```

A successful deletion will return no content, as follows:

Example Response

```
HTTP/1.1 204 No Content
Date: Fri, 19 Jan 2018 20:26:13 GMT
...
```

Email API

Introduction

The Email API exposes a set of endpoints for specifying and viewing SMTP server settings used by NXRM for sending [email](#)(see page 261).

Please see the *API* section (found in the *Administration* section under *System → API*) of your Nexus Repository Manager instance for detailed information about the *email* API including sample requests and responses.

Endpoints

NEW IN 3.19 The email REST API endpoints can be used to accomplish the following:

- Get email configuration
- Set email configuration
- Disable email configuration
- Verify email configuration

IQ Server API

Introduction

The IQ Server API exposes HTTP endpoints for creating, updating, deleting and viewing of IQ server connection details.

Please see the *API* section (found in the *Administration* section under *System → API*) of your Nexus Repository Manager instance for detailed information about the IQ API including sample requests and responses.

Endpoints

NEW IN 3.19 The IQ⁸³⁵ REST API endpoints can be used to accomplish the following:

- Get IQ server configuration
- Update IQ server configuration
- Disable IQ server configuration
- Enable IQ server configuration
- Verify IQ server connection

Licensing API

Introduction

The *Product Licensing* API exposes a set of endpoints for [License Management](#)(see page 280).

Please see the *API* section (found in the *Administration* section under *System → API*) of your Nexus Repository Manager instance for detailed information about the *Product Licensing* API including sample requests and responses.

Endpoints

NEW IN 3.19 The license REST API endpoints can be used to accomplish the following:

- Get license
- Upload new license
- Delete license

Lifecycle API

 These operations are intended to be used with the guidance of Sonatype support. Usage without supervision of Sonatype is not supported or recommended.

835 <https://help.sonatype.com/display/NXIQ>

Introduction

The Lifecycle API allow Nexus Repository Manager administrators to change the phase in which a NXRM instance is running. A lifecycle phase is a step in the start up process used to group similar components and ensure that their dependencies are started before them. In order of execution during startup, the phases are:

1. OFF - NXRM is completely off and the process is shut down.
2. KERNEL - The most basic parts of the node are running.
3. STORAGE - The databases and caches are setup.
4. RESTORE - Any restoring from backups happens in this phase.
5. UPGRADE - If any upgrades are needed, they will occur is this phase.
6. SCHEMAS - This creates any missing schemas in the database.
7. EVENTS - The event bus is initialized.
8. SECURITY - The parts of NXRM responsible for security are enabled.
9. SERVICES - This phase starts the all of the services, such as repositories.
10. CAPABILITIES - The capabilities are started; see [Accessing and Configuring Capabilities](#)⁸³⁶
11. TASKS - The task scheduler is enabled, and tasks may run.

Endpoints

Lifecycle Phase

```
GET /service/rest/v1/lifecycle/phase
```

This returns the current phase in which NXRM is running.

```
curl -u admin:admin123 -X GET "http://localhost:8081/service/rest/v1/lifecycle/phase" -H "accept: text/plain"
```

Example Response

```
TASKS
```

```
PUT /service/rest/v1/lifecycle/phase
```

⁸³⁶<https://help.sonatype.com/display/NXRM3M/System+Configuration#SystemConfiguration-AccessingandConfiguringCapabilities>

This sets NXRM to run in a particular phase. Any features that are enabled by later phases are shut down. e.g. Setting NXRM to the CAPABILITIES phase will stop any tasks from running, since they only occur in the TASKS phase.

```
curl -u admin:admin123 -X PUT "http://localhost:8081/service/rest/v1/lifecycle/phase" -H "accept: application/json" -H "Content-Type: text/plain" -d "SERVICES"
```

Lifecycle Bounce

```
PUT /service/rest/v1/lifecycle/bounce
```

This operation takes the instance down to the supplied Lifecycle phase, and then back to the phase in which it was originally running. This is used primarily to restart phase in one operation.

```
curl -u admin:admin123 -X PUT "http://localhost:8081/service/rest/v1/lifecycle/bounce" -H "accept: application/json" -H "Content-Type: text/plain" -d "TASKS"
```

Maintenance API

 Available in Nexus Repository Pro

Introduction

These REST endpoints allow the user to inspect and manipulate the state of a node in Orient database cluster. The expectation is that these will be used mainly for troubleshooting and recovery operations.

Possible Database Names: accesslog, component, config, security

Endpoints

Check Database

```
PUT /service/rest/v1/maintenance/{databaseName}/check
```

Checks database pages for corruption and checks that indices cover all records (i.e. no duplicates).

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/check
```

Example Response

```
{  
  "pageCorruption" : false,  
  "indexErrors" : 0  
}
```

Reinstall Database

 Only available in HA-C

```
PUT /service/rest/v1/maintenance/{databaseName}/reinstall
```

Attempts to reinstall the full database from the rest of the cluster.

 The user can't force a reinstall from a specific node, but typically Orient will choose the node that most recently took a backup or the oldest member (assuming that's not the node requesting the reinstall)

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/reinstall
```

Example Response

```
{  
  "reinstalled" : true  
}
```

Repair Database

```
PUT /service/rest/v1/maintenance/{databaseName}/repair
```

Attempts to repair any corrupt database pages and rebuilds indices.

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/repair
```

Example Response

```
{  
  "indexErrorsRemaining" : 0,  
  "repairedPages" : 0  
}
```

Database Role

 Only available in HA-C

This attribute represents the role of the database node in an Orient cluster. The role of MASTER represents a fully writable database node in an Orient cluster which participates in the writeQuorum. The role of REPLICA represents a database node in read-only mode which is accepting only idempotent commands, e.g. read and queries.

Possible Roles: MASTER, REPLICA

Get Role

```
GET /service/rest/v1/maintenance/{databaseName}/role
```

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/maintenance/component/role
```

Example Response

```
MASTER
```

Set Role

```
PUT /service/rest/v1/maintenance/{databaseName}/role
```

When setting a database node's role to REPLICA an override is added to Orient's configuration so that node will always be a REPLICA, regardless of the cluster's default role. When the database node is set back to MASTER that override is removed, so it will defer to the cluster's default role - which is MASTER unless it's been frozen.

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/role -H "Content-Type: text/plain" -d "MASTER"
```

Example Response

Database Status

 Only available in HA-C

This attribute represents the status of the database node in an Orient cluster. The status of ONLINE represents a database node in a normal state which is fully participating in the cluster. The status of OFFLINE represents a database node that is not currently participating in the cluster. The status of NOT_AVAILABLE represents a database node that is not currently available for some reason.

Possible Statuses: ONLINE, OFFLINE, NOT_AVAILABLE

Get Status

```
GET /service/rest/v1/maintenance/{databaseName}/status
```

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/maintenance/component/status
```

Example Response

```
ONLINE
```

Set Status

```
PUT /service/rest/v1/maintenance/{databaseName}/status
```

⚠ Setting the database status to NOT_AVAILABLE triggers an automatic delta-sync with the rest of the cluster. If this is successful, it will move back to ONLINE.

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/status -H "Content-Type: text/plain" -d "ONLINE"
```

Example Response

Nodes API

i Available in Nexus Repository Pro

Introduction

These endpoints allow us to get information about the nodes in the current cluster and update the friendly names of individual nodes. The list of nodes returned is a view of the state of the system and cannot be edited through this API. From this API, nodes can be assigned friendly names, but that is currently the only editable attribute. The friendly names may provide a more convenient mechanism to identify individual nodes for administrative activities.

Endpoints

Get Nodes

```
GET /service/rest/v1/nodes
```

This endpoint gives us a listing of the nodes currently in the cluster.

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/nodes
```

The information returned includes a unique nodeIdentity for each node along with the current socket address binding and the current friendly name of the node:

```
[ {  
    "nodeIdentity" : "20550283-C2239399-2DC0307C-DE78EE76-0670E0BC",  
    "socketAddress" : "/172.18.0.2:5701",  
    "friendlyName" : null  
, {  
    "nodeIdentity" : "2840754A-E8053498-0DA01A51-9A7582DD-A52D5279",  
    "socketAddress" : "/172.18.0.3:5701",  
    "friendlyName" : null  
, {  
    "nodeIdentity" : "A54F51B3-F0EBBD6F7-23D4C3A0-ABA922CD-9DCE73EA",  
    "socketAddress" : "/172.18.0.4:5701",  
    "friendlyName" : null  
} ]
```

Update Nodes

```
PUT /service/rest/v1/nodes
```

This endpoint allows us to update friendly names for nodes in the cluster:

```
curl -u admin:admin123 -X PUT --header 'Content-Type: application/json' \  
http://localhost:8081/service/rest/v1/nodes \  
-d '{"nodeIdentity" : "20550283-C2239399-2DC0307C-DE78EE76-0670E0BC", "socketAddress" : "/  
172.18.0.2:5701", "friendlyName" : "node1"}'
```

If successful, this operation returns the updated information for the node:

```
{  
    "nodeIdentity" : "20550283-C2239399-2DC0307C-DE78EE76-0670E0BC",  
    "socketAddress" : "/172.18.0.2:5701",  
    "friendlyName" : "node1"  
}
```

Read-Only API

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

Introduction

This set of endpoints is used to enable and disable read-only mode on the underlying database. This is useful when conducting some maintenance activities.

Endpoints

Get State

```
GET /service/rest/v1/read-only
```

This endpoint allows us to query the current read-only state of the database.

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/read-only
```

The resulting JSON document contains information about whether or not the database is currently "frozen" (i.e. in read-only mode), the reason why it may have been frozen, and whether or not the freeze was system initiated:

Example Response

```
{
  "frozen" : false,
  "summaryReason" : "",
  "systemInitiated" : false
}
```

Enable

```
POST /service/rest/v1/read-only/freeze
```

This endpoint allows us to put the database into read-only mode.

```
curl -u admin:admin123 -X POST http://localhost:8081/service/rest/v1/read-only/freeze
```

After issuing the freeze action, the database should be in read-only mode.

```
$ curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/read-only
{
  "frozen" : true,
  "summaryReason" : "activated by an administrator at 2018-01-22 15:04:05 +00:00",
  "systemInitiated" : false
}
```

Release

```
POST /service/rest/v1/read-only/release
```

This endpoint allows us to take the database out of read-only mode.

```
curl -u admin:admin123 -X POST http://localhost:8081/service/rest/v1/read-only/release
```

After issuing the release action, the database should no longer be in read-only mode.

```
$ curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/read-only
{
  "frozen" : false,
  "summaryReason" : "",
  "systemInitiated" : false
}
```

Force Release

```
POST /service/rest/v1/read-only/force-release
```

In certain rare instances we may need to forcibly take the database out of read-only mode. This endpoint allows us to do that.

```
curl -u admin:admin123 -X POST http://localhost:8081/service/rest/v1/read-only/force-release
```

After issuing the force-release action, the database should no longer be in read-only mode.

```
$ curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/read-only
{
  "frozen" : false,
  "summaryReason" : "",
  "systemInitiated" : false
}
```

Repositories API

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

This set of endpoints is for interacting with repositories directly.

Endpoints

List Repositories

```
GET /service/rest/v1/repositories
```

This endpoint allows us to iterate through a listing of repositories a user has browse access to.

Let's get a listing of the repositories that the user `admin` can browse:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/repositories'
```

This produces a response that is a listing of all repositories the user is allowed to browse:

Example Response

```
[  
  {  
    "name": "nuget.org-proxy",  
    "format": "nuget",  
    "type": "proxy",  
    "url": "http://localhost:8081/repository/nuget.org-proxy",  
    "attributes" : {  
      "proxy" : {  
        "remoteUrl" : "https://www.nuget.org/api/v2/"  
      }  
    }  
,  
  {  
    "name": "maven-releases",  
    "format": "maven2",  
    "type": "hosted",  
    "url": "http://localhost:8081/repository/maven-releases"  
},  
...  
]
```

This endpoint returns all repositories and does not allow for pagination.

Note that the ordering of repositories is consistent across multiple queries and is not alphabetical.

Search API

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

The Search API facilitates searching for components and assets in addition to downloading a specific asset.

Endpoints

Search Components

```
GET /service/rest/v1/search
```

This endpoint is useful for finding components based on general criteria (e.g. group, name, version) as well as format specific attributes (e.g. maven.baseVersion, npm.scope, yum.architecture). The search uses the same mechanism used by the Repository Manager UI in order to find components. Details about the criteria that can be used for searching are described in the generated documentation in Repository Manager under the administrative view System/API (`<nexus_url>/#admin/system/api`).

The search takes the form of a GET request against the endpoint which returns a JSON document with information about the components (and associated assets) that were found.

As an example, we can search for all components in the maven-central repository which have a group of `org.osgi`:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search?repository=maven-central&group=org.osgi'
```

This returns a JSON document containing a list of items that correspond to the components found during the search:

Example Response

```
{  
  "items" : [ {  
    "id" : "bWF2ZW4tY2VudHJhbDoyZTQ3ZGRhMGYxYjU1NWUwNzE10WRjOWY5ZGQzZmVmNA",  
    "repository" : "maven-central",  
    "format" : "maven2",  
    "group" : "org.osgi",  
    "name" : "org.osgi.core",  
    "version" : "4.3.1",  
    "assets" : [ {  
      "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/  
org.osgi.core-4.3.1-sources.jar",  
      "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1-sources.jar",  
      "id" : "bWF2ZW4tY2VudHJhbDp1MDE40GVkMDcyOGZhNjhNmNDExNzU20GU1MjQ2NjZiYg",  
      "repository" : "maven-central",  
      "format" : "maven2",  
      "checksum" : {  
        "sha1" : "80bfafcf783988442b3a58318face1d2132db33d",  
        "md5" : "87ee0258b79dc852626b91818316b9c3"  
      }  
    }, {  
      "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/  
org.osgi.core-4.3.1.jar",  
      "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.jar",  
      "id" : "bWF2ZW4tY2VudHJhbDpkMDY0ODA0YThlZDVhZDZlNjhZGU5MWNmM2NiZTgzMw",  
      "repository" : "maven-central",  
      "format" : "maven2",  
      "checksum" : {  
        "sha1" : "5458ffe2ba049e76c29f2df2dc3ffccddf8b839e",  
        "md5" : "8053bbc1b55d51f5abae005625209d08"  
      }  
    }, {  
      "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/  
org.osgi.core-4.3.1.pom",  
      "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.pom",  
      "id" : "bWF2ZW4tY2VudHJhbDo2NTRiYjdkMGE10TIxMzg10WZhMTVkMzNmYWU1ZmY30A",  
      "repository" : "maven-central",  
      "format" : "maven2",  
      "checksum" : {  
        "sha1" : "79391fc69dd72ad1fd983d01b4572f93f644882b",  
        "md5" : "3d87a59bcd4b131d9a63e87e0ed924a"  
      }  
    } ]  
  },  
  "continuationToken" : null  
}
```

In this case we've found one component which has three assets associated with it. Each component and asset has an id that can be used with the component and asset specific endpoints. Since only one item was found, there is no need for pagination so the continuationToken has a value of null to signify that this is the last page of items. The downloadUrl can be used to directly download the asset from the repository manager.

This endpoint uses a [pagination](#)⁸³⁷ strategy that can be used to iterate through all the search results if desired.

Search Assets

```
GET /service/rest/v1/search/assets
```

This endpoint is focused on searching for assets. All of the same search criteria are available as in the component search above, but only assets will be returned.

Let's again search the maven-central repository for assets which have a group of org.osgi:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search/assets?repository=maven-central&group=org.osgi'
```

This returns a JSON document containing a list of items corresponding to the assets found during the search:

⁸³⁷ <https://help.sonatype.com/display/NXRM3M/Pagination>

Example Response

```
{  
  "items" : [ {  
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/  
org.osgi.core-4.3.1-sources.jar",  
    "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1-sources.jar",  
    "id" : "bWF2ZW4tY2VudHJhbDplMDE4OGVkJDcyOGZhNjhNDEXNzU20GU1MjQ2NjZiYg",  
    "repository" : "maven-central",  
    "format" : "maven2",  
    "checksum" : {  
      "sha1" : "80bfafcf783988442b3a58318face1d2132db33d",  
      "md5" : "87ee0258b79dc852626b91818316b9c3"  
    }  
  }, {  
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/  
org.osgi.core-4.3.1.jar",  
    "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.jar",  
    "id" : "bWF2ZW4tY2VudHJhbDpkMDY0ODA0YThlZDVhZDZlNjhZGU5MWNmM2NiZTgzMw",  
    "repository" : "maven-central",  
    "format" : "maven2",  
    "checksum" : {  
      "sha1" : "5458ffe2ba049e76c29f2df2dc3ffccddf8b839e",  
      "md5" : "8053bbc1b55d51f5abae005625209d08"  
    }  
  }, {  
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/  
org.osgi.core-4.3.1.pom",  
    "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.pom",  
    "id" : "bWF2ZW4tY2VudHJhbDo2NTRiYjdkMGE1OTIxMzg1OWZhMTVkJMzNmYWU1ZmY30A",  
    "repository" : "maven-central",  
    "format" : "maven2",  
    "checksum" : {  
      "sha1" : "79391fc69dd72ad1fd983d01b4572f93f644882b",  
      "md5" : "3d87a59bcdb4b131d9a63e87e0ed924a"  
    }  
  } ],  
  "continuationToken" : null  
}
```

This shows us that we found three assets and that there are no additional pages of results.

This endpoint uses a [pagination](#)⁸³⁸ strategy that can be used to iterate through all the search results if desired.

Search and Download Asset

```
GET /service/rest/v1/search/assets/download
```

⁸³⁸ <https://help.sonatype.com/display/NXRM3M/Pagination>

This endpoint is specifically designed to search for one asset and then redirect the request to the downloadUrl of that asset.

Let's say we want to download the asset corresponding to a jar whose group is

org.osgi
, name is
org.osgi.core
, and version is
4.3.1

To achieve this, it is necessary that the search returns a single asset. Continuing our example, we can use the asset search endpoint above to refine our search until it returns a single asset:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search/assets?group=org.osgi&name=org.osgi.core&version=4.3.1&maven.extension=jar&maven.classifier'
```

Note that in the above example, the last parameter "maven.classifier" has no value. This indicates that the matched asset should have no classifier.

The response returned by the above looks like this:

Example Response

```
{
  "items" : [ {
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.jar",
    "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.jar",
    "id" : "bWF2ZW4tY2VudHJhbDpkMDY00DA0YThlZDVhZDZ1NjhZGU5MWNmM2NiZTgzMw",
    "repository" : "maven-central",
    "format" : "maven2",
    "checksum" : {
      "sha1" : "5458ffe2ba049e76c29f2df2dc3ffccddf8b839e",
      "md5" : "8053bbc1b55d51f5abae005625209d08"
    }
  }],
  "continuationToken" : null
}
```

Then we can append

/download

to produce the URL we can use to download the asset:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search/assets/download?group=org.osgi&name=org.osgi.core&version=4.3.1&maven.extension=jar&maven.classifier'
```

If successful, this will give us a 302 response and redirect us to the repository manager download URL for the asset.

Example Response

```
HTTP/1.1 302 Found
Content-Length: 0
Date: Fri, 19 Jan 2018 17:34:21 GMT
Location: http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/
org.osgi.core-4.3.1.jar
...
```

Note that the search parameters in the example contain a parameter (`maven.classifier`) with no value:

```
...?group=org.osgi&name=org.osgi.core&version=4.3.1&maven.extension=jar&maven.classifier
```

Specifying a parameter with no value is an indicator to the search endpoint to only return assets which correspondingly have no value for that parameter. This technique can be used filter down search results to a single asset for some formats such as Maven where the component contains multiple assets that match all search criteria specified right up to the extension. For example, a Maven component can have the actual JAR file for a library, plus have a sources JAR. In these cases, just specifying `maven.extension=jar` is not specific enough to return a single asset but we can further narrow the search results by specifying that we want only the main JAR file asset not the sources JAR by including an empty `maven.classifier` parameter.

Alternatively, let's now assume that instead of wanting the main `org.osgi.core` JAR we desire to search for and download the sources JAR (e.g. `org.osgi.core-4.3.1-sources.jar`). That is, we want to search for and download the asset corresponding to a jar whose group is `org.osgi`, name is `org.osgi.core`, version is 4.3.1 and `maven.classifier` is `sources`.

We can achieve this by using the asset search endpoint and including the `maven.classifier` with a specified value of `sources`:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search/assets/download?
group=org.osgi&name=org.osgi.core&version=4.3.1&maven.classifier=sources'
```

If successful, this will give us a 302 response and redirect us to the repository manager download URL for the asset.

Example Response

```
HTTP/1.1 302 Found
Content-Length: 0
Date: Fri, 19 Jan 2018 17:34:21 GMT
Location: http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1-
sources.jar
...
```

Downloading the Latest Version of an Asset

NEW IN 3.16.0

You can download the latest version of an asset by using the "sort=version" parameter. For example, this will download the highest semantic version available for the Maven group ID and artifact ID specified:

```
curl -X GET "http://localhost:8081/service/rest/v1/search/assets/download?sort=version&repository=maven-snapshots&maven.groupId=org.foo.bar&maven.artifactId=project&maven.extension=jar" -H "accept: application/json"
```

To get the latest version of a Maven snapshot, also include the base version:

```
curl -X GET "http://localhost:8081/service/rest/v1/search/assets/download?sort=version&repository=maven-snapshots&maven.groupId=org.foo.bar&maven.artifactId=project&maven.baseVersion=1.2.3-SNAPSHOT&maven.extension=jar" -H "accept: application/json"
```

Security Management API

Please see the *API* documentation (found in the *Administration* section under *System*) of your Nexus Repository Manager instance for complete documentation about the specific endpoints available in your local server.

Content Selectors

NEW IN 3.19 The content selectors REST API endpoints can be used to create and manage [content selectors](#)(see page 251):

- list the content selectors
- create or modify content selectors

LDAP

NEW IN 3.19 The [LDAP](#)(see page 324) endpoints can be used to accomplish the following:

- List all LDAP servers
- Create LDAP server
- Retrieve the details of a single LDAP server
- Update LDAP server
- Delete LDAP server
- Change LDAP server ordering

Privileges

NEW IN 3.19 The privileges REST API endpoints can be used to create and manage [privileges](#)(see page 313):

- list the privileges
- create or modify privileges

Roles

NEW IN 3.19 The roles REST API endpoints can be used to create and manage [roles](#)(see page 317) and their permissions within NXRM:

- list the roles from any configured user source (internal or external)
- create or modify local roles and external role mappings

Users

NEW IN 3.17 The users REST API endpoints can be used to create and manage [users](#)(see page 320) and their permissions within NXRM:

- search for users available to NXRM whether they were defined locally or from a configured authentication source such as LDAP
- create or modify local users
- change the roles associated with an external user
- reset an individual [user token](#)(see page 335)
- reset all user tokens in the system

User Sources

NEW IN 3.17 This endpoint provides a list of the available users sources in the Nexus Repository Manager, these are used in other REST endpoints to indicate the source of certain types of entities, e.g. a user from an LDAP server.

Script API

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

Introduction

This is a powerful scripting API that provides methods to simplify provisioning and executing other complex tasks in the repository manager. These APIs can be invoked from scripts that are published to the repository manager and executed within the application server.

More detailed documentation is available in the following topics:

- [Writing Scripts](#)(see page 502)
- [Managing and Running Scripts](#)(see page 504)
- [Examples](#)(see page 506)

Endpoints

Add Script

```
POST /service/rest/v1/script
```

This endpoint allows us to create a utility script within the repository manager.

As an example, let's create a simple script that simply logs "Hello, World!":

```
curl -u admin:admin123 -X POST --header 'Content-Type: application/json' \
http://localhost:8081/service/rest/v1/script \
-d @helloWorld.json
```

The JSON document we send includes the unique name of the script, the contents of the script itself, and the language type of the script (in this case groovy).

helloWorld.json

```
{
  "name": "helloWorld",
  "content": "log.info('Hello, World!')",
  "type": "groovy"
}
```

If you are using Windows and want to use Powershell to add a script you can use the following commands:

```
$bytes = [System.Text.Encoding]::UTF8.GetBytes("admin:admin123")
$cred = [System.Convert]::ToBase64String($bytes)
$uri = "http://localhost:8081/service/rest/v1/script"
$jsonPath = "c:\test\helloWorld.json"
Invoke-RestMethod -Uri $uri -Method POST -Headers @{"Authorization"="Basic $cred"} -ContentType "application/json" -InFile $jsonPath
```

List Scripts

```
GET /service/rest/v1/script
```

This endpoint allows us to get a listing of all the scripts currently defined in the repository manager.

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/script
```

Building on the example from above, we only have the one script that we just created:

Example Response

```
[ {
  "name" : "helloWorld",
  "content" : "log.info('Hello, World!')",
  "type" : "groovy"
} ]
```

Get Script

```
GET /service/rest/v1/script/{name}
```

This endpoint allows us to get the details for an individual script by its name.

Let's get the details for the helloWorld script that we defined earlier:

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/script/helloworld
```

Example Response

```
{  
  "name" : "helloWorld",  
  "content" : "log.info('Hello, World!')",  
  "type" : "groovy"  
}
```

Update Script

```
PUT /service/rest/v1/script/{name}
```

This endpoint allows us to update the contents of an existing script.

Let's update our script to log "Hello, Nexus!" instead of "Hello, World!":

```
curl -u admin:admin123 -X PUT --header 'Content-Type: application/json' \  
http://localhost:8081/service/rest/v1/script/helloWorld \  
-d @helloNexus.json
```

helloNexus.json

```
{  
  "name": "helloWorld",  
  "content": "log.info('Hello, Nexus!')",  
  "type": "groovy"  
}
```

Run Script

```
POST /service/rest/v1/script/{name}/run
```

This endpoint allows us to run a script that we have already defined in the repository manager.

Let's run our helloWorld script:

```
curl -u admin:admin123 -X POST --header 'Content-Type: text/plain' \  
http://localhost:8081/service/rest/v1/script/helloWorld/run
```

The JSON response tells us which script ran and the returned result from the script:

Example Response

```
{  
  "name" : "helloWorld",  
  "result" : "null"  
}
```

We should also see something like the following in the console log:

```
2018-01-22 16:28:55,495+0000 INFO [qtp74765809-267] admin  
org.sonatype.nexus.script.plugin.internal.rest.ScriptResource$$EnhancerByGuice$$99ce415 - Hello, Nexus!
```

Privileges

To run a script, a user needs to have the nx-script-\${script-name}-run privilege. A user with the nx-script-*-run privilege can run all scripts. Note that the will also need privileges assigned to them which allow them to perform the actions that the script does.

Delete Script

```
DELETE /service/rest/v1/script/{name}
```

This endpoint allows us to delete scripts once they are no longer needed.

Let's delete our helloWorld script:

```
curl -u admin:admin123 -X DELETE http://localhost:8081/service/rest/v1/script/helloworld
```

A successful deletion returns no content:

Example Response

```
HTTP/1.1 204 No Content  
Date: Mon, 22 Jan 2018 22:58:03 GMT  
...
```

Writing Scripts

The scripting language used on the repository manager is [Groovy](#)⁸³⁹. Any editor can be used to author the scripts.

The available APIs are contained in a number of JAR files. All these files, including JavaDoc and Sources archives, are available from the [Central Repository](#)⁸⁴⁰. They can be manually downloaded and extracted. E.g. the different versions and the specific JAR files for org.sonatype.nexus:nexus-core are available in versioned directories at <http://repo1.maven.org/maven2/org/sonatype/nexus/nexus-core/>.

This manual process can be simplified and improved by the usage of a Maven project declaring the relevant components as dependencies. An example project with this setup called `nexus-script-example` and a few scripts are available in the [example project](#)⁸⁴¹.

Maven Project pom.xml Declaring the API Dependencies for Scripting

⁸³⁹ <http://www.groovy-lang.org/>

⁸⁴⁰ <http://search.maven.org/>

⁸⁴¹ <https://github.com/sonatype/nexus-book-examples/tree/nexus-3.x>

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example.automation</groupId>
    <artifactId>nexus-script-demo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <nx-version>3.3.0-01</nx-version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.sonatype.nexus</groupId>
            <artifactId>nexus-core</artifactId>
            <version>${nx-version}</version>
        </dependency>
        <dependency>
            <groupId>org.sonatype.nexus</groupId>
            <artifactId>nexus-script</artifactId>
            <version>${nx-version}</version>
        </dependency>
        <dependency>
            <groupId>org.sonatype.nexus</groupId>
            <artifactId>nexus-security</artifactId>
            <version>${nx-version}</version>
        </dependency>
        <dependency>
            <groupId>org.sonatype.nexus.plugins</groupId>
            <artifactId>nexus-script-plugin</artifactId>
            <version>${nx-version}</version>
        </dependency>
    </dependencies>
</project>
```

Development environments such as IntelliJ IDEA or Eclipse IDE can download the relevant JavaDoc and Sources JAR files to ease your development. Typically you would create your scripts in `src/main/groovy` or `src/main/scripts`.

The scripting API exposes specific tooling for IntelliJ IDEA that allows you to get access to code completion and similar convenience features, while writing your scripts in this Maven project. Currently the API exposes four main providers with numerous convenient methods:

- `core`
- `repository`
- `blobStore`
- `security`

The API is deliberately designed to be simple to use. It encapsulates complex configuration in single method invocations. Many of the included methods use default values that can be omitted. For example, the method to create a hosted repository using the Maven format in the simplest usage simply requires a name.

```
repository.createMavenHosted("private")
```

This method simply uses the default values for the rest of the parameters and is therefore equivalent to:

```
repository.createMavenHosted("private", BlobStoreManager.DEFAULT_BLOBSTORE_NAME, true,  
VersionPolicy.RELEASE, WritePolicy.ALLOW_ONCE, LayoutPolicy.STRICT)
```

You can inspect the default values in the API documentation available by inspecting the declaration of the specific methods in your IDE or by viewing the JavaDoc.

In terms of overall complexity of the scripts created, it is best to break large tasks up into multiple scripts and therefore invocations.

Managing and Running Scripts

Once you have completed the creation of your script, you need to publish it to the repository manager for execution. This is done by REST API invocations.

For Repository Manager version 3.8.x and onwards, the REST API endpoint is:

```
http://localhost:8081/service/rest/v1/script
```

For Repository Manager versions pre 3.8.x, the REST API endpoint is:

```
http://localhost:8081/service/siesta/rest/v1/script
```

 **Note**

The following examples given below are against NXRM version 3.8+ so please amend the endpoint accordingly if running any NXRM 3 version before that.

This endpoint accepts JSON-formatted payloads with your script as the content.

Example JSON formatted file `maven.json` with a simple repository creation script:

```
{  
  "name": "maven",  
  "type": "groovy",  
  "content": "repository.createMavenHosted('private')"  
}
```

The JSON file `maven.json` located in the current directory can be published to the repository manager with an HTTP POST like:

```
curl -v -X POST -u admin:admin123 --header "Content-Type: application/json" 'http://localhost:8081/service/rest/v1/script' -d @maven.json
```

A list of scripts stored on the repository manager can be accessed with:

```
curl -v -X GET -u admin:admin123 'http://localhost:8081/service/rest/v1/script'
```

The same call with a script name appended returns the actual script content.

A script can be executed by sending a POST to the run method of the specific script.

```
curl -v -X POST -u admin:admin123 --header "Content-Type: text/plain" 'http://localhost:8081/service/rest/v1/script/maven/run'
```

A successful execution should result in a HTTP/1.1 200 OK.

Scripts can be removed with a HTTP DELETE operation to the specific script:

```
curl -v -X DELETE -u admin:admin123 'http://localhost:8081/service/rest/v1/script/maven'
```

Parameters

Scripts can receive run-time parameters via the REST API:

```
curl -v -X POST -u admin:admin123 --header "Content-Type: text/plain" 'http://localhost:8081/service/rest/v1/script/updateAnonymousAccess/run' -d 'false'
```

and receive them as arguments that have to be parsed by the script as desired:

```
security.setAnonymousAccess(Boolean.valueOf(args))
```

Multiple parameters can be passed in via a JSON file in the POST request:

params.json:

```
{  
  "repoName": "snapshots",  
  "groupId": "org.some.project",  
  "artifactId": "someartifact"  
}
```

Then the command would be similar to the following:

```
curl -k -u user:pass -X POST --header 'Content-Type: text/plain' http://localhost:8081/service/rest/v1/  
script/rebuild-maven-metadata/run -d @params.json
```

And you can access the parameters in your script code:

```
def request = new JsonSlurper().parseText(args);  
assert request.repoName: 'repoName parameter is required';  
assert request.groupId: 'groupId parameter is required';  
assert request.artifactId: 'artifactId parameter is required';
```

Interaction with the REST API for scripts can be done with any scripting language capable of HTTP calls as mentioned above. In the following section you can find some further detailed examples.

Examples

The API for scripts is capable of a number of different tasks. This section provides examples for script writing, publishing and executing them.

The **simple-shell-example** project in the [scripting section of the example project](#)⁸⁴² includes a number of JSON file with simple scripts:

maven.json

simplest script to create a hosted Maven repository

npm.json

simple script to create a hosted and proxy repository as well as a repository group for npm usage

bower.json

⁸⁴² <https://github.com/sonatype-nexus-community/nexus-scripting-examples>

simple script to create a hosted and proxy repository as well as a repository group for bower usage

anonymous.json

parameterized script to enable or disable anonymous access

Simple shell scripts are added to contain the curl invocations to manage scripts via the REST API:

create.sh

Upload a specified JSON file

delete.sh

Delete a script specified by its name

list.sh

List all deployed scripts

run.sh

Run a script specified by its name

setAnonymous.sh

Run the anonymous script on the server with the parameter true or false

update.sh

Update an existing script by specifying the name and the JSON file to use for the update

An example sequence of creating and running a script is:

```
./create.sh maven.json  
./run.sh maven
```

Subsequently you could list all scripts and delete the maven script with:

```
./list.sh  
./delete.sh maven
```

Since scripts are typically longer than a single line and creating them in a separate file in the IDE is recommended, using a helper script that formats a .groovy file into a JSON file and submits it to the repository manager can be a convenient approach.

The **complex-script** project in the [scripting section of example project](#)⁸⁴³ includes an example implementation using Groovy invoked from a shell script. All scripts in this folder can be published and executed via the provision.sh file. This results in the download of all required dependencies and the upload and execution of the referenced script. Alternatively, you can provision the scripts individually:

⁸⁴³ <https://github.com/sonatype-nexus-community/nexus-scripting-examples>

```
groovy addUpdateScript.groovy -u "admin" -p "admin123" -n "raw" -f "rawRepositories.groovy" -h "http://localhost:8081"  
curl -v -X POST -u admin:admin123 --header "Content-Type: text/plain" "http://localhost:8081/service/rest/v1/script/raw/run"
```

The following scripts are available:

npmAndBowerRepositories.groovy

for NPM and Bower repositories suitable for server-side and client JavaScript-based development

rawRepositories.groovy

creates a new blob store and uses it for a hosted raw repository

security.groovy

disables anonymous access, creates a new administrator account, creates a new role with a simple expansion to anonymous user role and a user, creates a new role with publishing access to all repositories and a user

core.groovy

configures the base URL capability and a proxy server

Logging from your scripts into the repository manager logs is automatically available and performed with the usual calls:

```
log.info('User jane.doe created')
```

The result of the last script line is by default returned as a string. This can be a message as simple as Success! or more complex structured data.

For instance, you can easily return JSON using built-in Groovy classes like:

```
return groovy.json.JsonOutput.toJson([result: 'Success!'])
```

which looks like:

```
{  
    "result": "Success!"  
}
```

Passing parameters to the script can use JSON encoded arguments like:

```
{  
    "id": "foo",  
    "name": "bar",  
    "description": "baz",  
    "privilegeIds": ["nx-all"],  
    "roleIds": ["nx-admin"]  
}
```

which in turn can be parsed using the `JsonSlurper` class in the script:

```
import groovy.json.JsonSlurper  
  
//expects json string with appropriate content to be passed in  
def role = new JsonSlurper().parseText(args)  
  
security.addRole(role.id, role.name, role.description, role.privilegeIds, role.roleIds)
```

You can read more about how to work with XML and JSON with Groovy on <http://groovy-lang.org/processing-xml.html> and <http://groovy-lang.org/json.html>.

Status API

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

Introduction

This set of endpoints is used to determine the status of an instance. This is useful when conducting some maintenance activities.

Endpoints

Status

```
GET /service/rest/v1/status
```

This endpoint allows us to query if the instance can successfully serve read requests.

Curl Example

```
curl -X GET http://localhost:8081/service/rest/v1/status
```

The response code contains the information. A 200 indicates the instance can serve read requests, a 503 otherwise.

Writable

```
GET /service/rest/v1/status/writable
```

This endpoint allows us to query if the instance can successfully serve write and read requests.

Curl Example

```
curl -X GET http://localhost:8081/service/rest/v1/status/writable
```

The response code contains the information. A 200 indicates the instance can serve write and read requests, a 503 otherwise.

Support API

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

Introduction

This endpoint allows us to download a support zip from an instance.

Endpoints

Supportzip

```
GET /service/rest/v1/support/supportzip
```

This endpoint allows us to download a support zip from an instance.

Curl Example

```
curl -u admin:admin123 -X POST "http://localhost:8081/service/rest/v1/support/supportzip" -H "accept: application/octet-stream" -H "Content-Type: application/json" -d "{ \"systemInformation\": true, \"threadDump\": true, \"metrics\": true, \"configuration\": true, \"security\": true, \"log\": true, \"taskLog\": true, \"auditLog\": true, \"jmx\": true, \"limitFileSizes\": true, \"limitZipSize\": true}" --output supportzip.zip
```

The response body contains the support zip.

Tasks API

 Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

This set of endpoints allows us to interact with tasks that have been created in the repository manager administrative UI.

Endpoints

List Tasks

```
GET /service/v1/tasks
```

This endpoint allows us to iterate through a listing of all the existing tasks.

Let's get a listing of the current tasks:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/tasks'
```

Example Response

```
{  
  "items" : [ {  
    "id" : "3f0173a8-c379-44c3-8ba3-2f0d0290bbfa",  
    "name" : "Rebuild all browse trees",  
    "type" : "create.browse.nodes",  
    "message" : null,  
    "currentState" : "WAITING",  
    "lastRunResult" : null,  
    "nextRun" : null,  
    "lastRun" : null  
, {  
    "id" : "dbb1f322-907a-4424-ad82-8d0a89deabe1",  
    "name" : "Rebuild all search indices",  
    "type" : "repository.rebuild-index",  
    "message" : null,  
    "currentState" : "WAITING",  
    "lastRunResult" : null,  
    "nextRun" : null,  
    "lastRun" : null  
, {  
    "id" : "07fa0c5d-8217-45da-91d3-fdfe9452e21f",  
    "name" : "Publish all maven indices",  
    "type" : "repository.maven.publish-dotindex",  
    "message" : null,  
    "currentState" : "WAITING",  
    "lastRunResult" : null,  
    "nextRun" : null,  
    "lastRun" : null  
, {  
    "id" : "2c8fd1f8-2395-4576-a172-a68089bb2ef7",  
    "name" : "Compact default blob store",  
    "type" : "blobstore.compact",  
    "message" : null,  
    "currentState" : "WAITING",  
    "lastRunResult" : null,  
    "nextRun" : null,  
    "lastRun" : null  
, {  
    "id" : "0261aed9-9f29-447b-8794-f21693b1f9ac",  
    "name" : "Hello World",  
    "type" : "script",  
    "message" : null,  
    "currentState" : "WAITING",  
    "lastRunResult" : null,  
    "nextRun" : null,  
    "lastRun" : null  
} ],  
  "continuationToken" : null  
}
```

This endpoint uses a [pagination](#)⁸⁴⁴ strategy that can be used to iterate through all the tasks if desired.

In this case, we only have one page of results as signified by the `null` continuationToken.

Get Task

```
GET /service/v1/tasks/{id}
```

This endpoint allows us to get the details about an individual task.

Let's take a look at the details for the "Hello World" task (i.e. `id=0261aed9-9f29-447b-8794-f21693b1f9ac`):

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/tasks/0261aed9-9f29-447b-8794-f21693b1f9ac'
```

Example Response

```
{
  "id" : "0261aed9-9f29-447b-8794-f21693b1f9ac",
  "name" : "Hello World",
  "type" : "script",
  "message" : null,
  "currentState" : "WAITING",
  "lastRunResult" : null,
  "nextRun" : null,
  "lastRun" : null
}
```

Run Task

```
POST /service/v1/tasks/{id}/run
```

This endpoint allows us to run an individual task.

For example, to run the "Hello World" task from above:

```
curl -u admin:admin123 -X POST 'http://localhost:8081/service/rest/v1/tasks/0261aed9-9f29-447b-8794-f21693b1f9ac/run'
```

⁸⁴⁴ <https://help.sonatype.com/display/NXRM3M/Pagination>

Example Response

```
HTTP/1.1 204 No Content
Date: Mon, 22 Jan 2018 22:19:47 GMT
...
```

Stop Task

```
POST /service/v1/tasks/{id}/stop
```

This endpoint allows us to stop an individual task. This is the equivalent of cancelling a task in the repository manager UI. Note that not all tasks will respond to a cancellation request.

For example to stop the "Hello World" task:

```
curl -u admin:admin123 -X POST 'http://localhost:8081/service/rest/v1/tasks/0261aed9-9f29-447b-8794-f21693b1f9ac/stop'
```

In this example the "Hello World" task was not running when the stop request was made. We will get a 409 response code that signifies that the requested task was not currently running:

Example Response

```
HTTP/1.1 409 Conflict
Content-Length: 0
Date: Mon, 22 Jan 2018 22:22:33 GMT
...
```

Pagination

Many of the REST API's make use of a pagination strategy for dealing with operations that can return a large number of items. This strategy is built around the notion of a `continuationToken` and a page size that determines the maximum number of items that can be returned in a single response.

When a `continuationToken` is present in a response and has a non-null value, this signifies that there are more items available:

Example Request

```
GET /service/rest/v1/<api>?<query>
```

Example Response

```
{  
  "items" : [  
    ...  
  ],  
  "continuationToken" : "88491cd1d185dd136f143f20c4e7d50c"  
}
```

The API that produced the response will take a `continuationToken` as an additional argument to the original query to signify that the next page of results is desired:

Example Request (next page)

```
GET /service/rest/v1/<api>?<query>&continuationToken=88491cd1d185dd136f143f20c4e7d50c
```

If this response also contains a non-null `continuationToken`, then its value can again be added to the original query to get the next page. This continues until the response returns a `continuationToken` with a value of `null` which signifies that there are no more pages of results.

Webhooks

(i) Available in Nexus Repository OSS and Nexus Repository Pro

Webhooks are defined as an HTTP callback. In simple terms, webhooks allow Nexus Repository Manager administrators to configure HTTP-based callbacks to notify external services of important events happening within Nexus Repository Manager.

Organizations are using webhooks in many ways to further automate their workflows, and integrate third party systems with key services. The Nexus Repository Manager provides the ability to use webhooks for global auditing, repository based events, and for events on specific repositories.

Using Webhooks

Setting up webhooks in Nexus Repository Manager can be accomplished by any user with sufficient privilege to create *Capabilities*, generally an administrator.

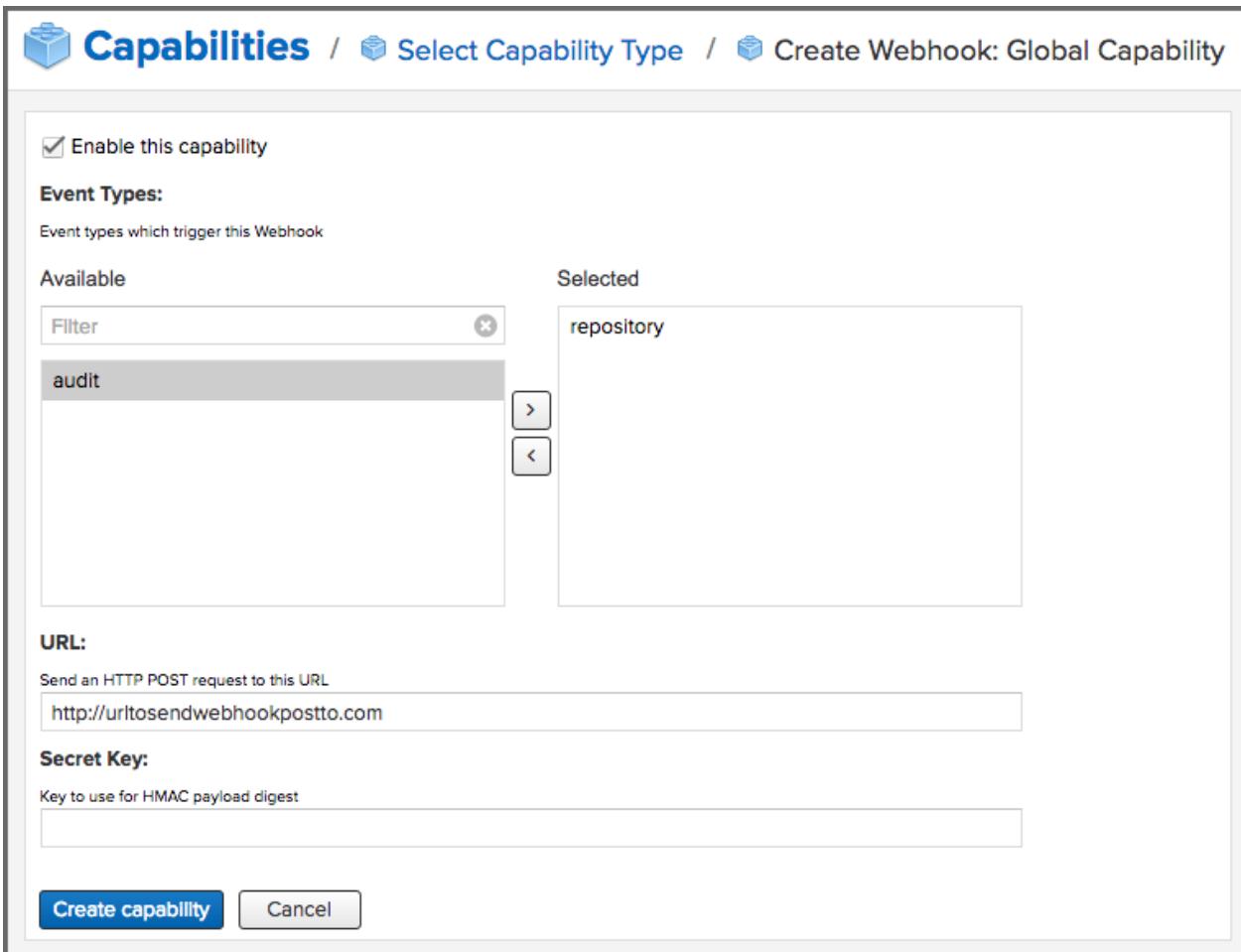
For ease of testing Webhooks, you can use a service such as [RequestBin](https://requestbin.net/)⁸⁴⁵ to quickly test out your desired Webhook. Alternatively any lightweight server that can allow you to see the contents of an HTTP POST will work.

⁸⁴⁵ <https://requestbin.net/>

Topics in this section:

- [Enabling A Global Webhook Capability\(see page 516\)](#)
- [Enabling A Repository Webhook Capability\(see page 518\)](#)
- [Working With HMAC Payloads\(see page 519\)](#)
- [Example Headers And Payloads\(see page 520\)](#)

Enabling A Global Webhook Capability



The screenshot shows the 'Create Webhook: Global Capability' configuration dialog. At the top, there is a checkbox labeled 'Enable this capability' which is checked. Below it, under 'Event Types:', there is a section titled 'Available' containing a list of event types: 'audit'. To the right, under 'Selected', is the word 'repository'. There are two buttons between the lists: a blue arrow pointing right and a red arrow pointing left. Below the event type selection, there is a section titled 'URL:' with a placeholder 'Send an HTTP POST request to this URL' and a text input field containing 'http://urltosendwebhookpostto.com'. Under 'Secret Key:', there is a placeholder 'Key to use for HMAC payload digest' with a corresponding input field. At the bottom of the dialog are two buttons: a blue 'Create capability' button and a grey 'Cancel' button.

Figure: Global Webhook: Management Capability Settings

To enable a global webhook, perform the following steps:

1. Select *Capabilities* to open the *Capabilities* panel, located in the *Administration* menu
2. Click the *Create capability* button to get to the *Select Capability Type* table
3. Select *Webhook: Global* to open the *Create Webhook: Global Capability* panel

4. Complete the form by selecting which *Event Types* you'd like to have events sent, entering the URL you would like the events to be sent to.

⚠ A *Global: Webhook* capability for *Event Type* audit requires that you also enable the *Audit* capability, in order for events to be fed to the webhook.

This form also includes an option to use a *Secret Key* for sending your events with a HMAC payload digest.

⚠ Webhooks will provide a `sha1` hash of the JSON body using a shared secret key. This allows you to verify the data integrity as well as its authenticity. The JSON body is hashed without whitespace and this hash is provided in the header as `X-Nexus-Webhook-Signature`.

After you enable the capability, you will now see the *Webhook: Global* capability in the list of *Capabilities*.

Enabling A Repository Webhook Capability

The screenshot shows the 'Capabilities' section of the Nexus Repository Manager 3 interface. At the top, there are three tabs: 'Capabilities' (selected), 'Select Capability Type', and 'Create Webhook: Repository Cap...'. Below the tabs, there is a checkbox labeled 'Enable this capability' which is checked. Under 'Repository:', a dropdown menu is set to 'maven-central'. In the 'Event Types:' section, there are two columns: 'Available' and 'Selected'. The 'Available' column contains 'asset' with a filter input field and two navigation buttons ('>' and '<'). The 'Selected' column contains 'component'. Under 'URL:', there is a text input field containing 'http://urltosendwebhookto.com'. Under 'Secret Key:', there is a text input field containing a single character '|'. At the bottom of the form are two buttons: 'Create capability' (highlighted in blue) and 'Cancel'.

Figure: Repository Webhook: Management Capability Settings

To enable a Webhook for a specific repository, perform the following steps:

1. Select *Capabilities* to open the *Capabilities* panel, located in the *Administration* menu
2. Click the *Create capability* button to get to the *Select Capability Type* table
3. Select *Webhook: Repository* to open the *Create Webhook: Repository Capability* panel
4. Complete the form by selecting which *Repository* you'd like to receive events from, select *Event Types* you'd like to have events sent for, and entering the URL you would like the events to be sent to

This form also includes an option to use a *Secret Key* for sending your events with a HMAC payload digest.

⚠ Nexus Repository Manager Webhooks will provide a sha1 hash of the JSON body using a shared secret key. This allows you to verify the data integrity as well as its authenticity. The JSON body is hashed without whitespace and this hash is provided in the header as X-Nexus-Webhook-Signature.

After you enable the capability, you will now see the *Webhook: Repository* capability in the list of *Capabilities*.

Working With HMAC Payloads

If you have enabled a secret key to generate an HMAC digest, a special header will be sent with all of your Webhook payloads. This header is X-Nexus-Webhook-Signature. It can be used to ensure that the message you receive is in fact what was originally generated.

For ease of getting you up and running with webhooks using HMAC, here is an example express based node.js script that can be used to verify that the payload you receive is what was originally sent.

app.js

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const crypto = require('crypto');
const secretKey = 'mysecretkey';

app.use(bodyParser.json());

app.post('/', function(req, res) {
  const body = req.body;
  const signature = req.headers['x-nexus-webhook-signature'];
  var hmacDigest = crypto.createHmac("sha1",
    secretKey).update(JSON.stringify(body)).digest("hex");

  console.log('Webhook received');
  console.log('Headers: ' + JSON.stringify(req.headers));
  console.log('Body: ' + JSON.stringify(req.body));
  console.log('HmacDigest: ' + hmacDigest);
  console.log('Signature: ' + signature);
  res.send();
});

app.listen(3000, function() {
  console.log('Server running on port 3000.');
});
```

This script can also be used for testing as an alternative to [RequestBin](#)⁸⁴⁶.

Example Headers And Payloads

To work with a webhook, you need to know a bit about the payload you'll be receiving. Each event you receive a payload for will contain special headers that describe what the event is.

Example Headers
X-Request-Id: d535c62b-063e-4ace-f7b579d6c37c Content-Type: application/json; charset=UTF-8 User-Agent: Nexus/3.1.0-SNAPSHOT (PRO; Mac OS X; 10.11.1; x86_64; 1.8.0_60) X-Nexus-Webhook-Signature: 687f3719b87232cf1c11b3ef7ea10c49218b6df1 X-Nexus-Webhook-Id: rm:repository:asset X-Nexus-Webhook-Delivery: 7f4a6dde-5c68-4999-bcc0-a62f3fb8ae48

Of special importance are the following three headers:

Header	Description
X-Webhook-Signature	This is the HMAC digest of the body of the payload, if an optional secret key has been configured
X-Nexus-Webhook-Delivery	This is a unique UUID identifying the event
X-Nexus-Webhook-Id	This is the event type, e.g. rm:repository:asset

Table 19.1. Event Specific Headers

A payload will be returned with each event type, an example of one for a repository asset webhook is shown below:

Example Payload
{ "timestamp" : "2016-11-10T23:57:49.664+0000", "nodeId" : "52905B51-085CCABB-CEBBEAAD-16795588-FC927D93", "initiator" : "admin/127.0.0.1", "repositoryName" : "npm-proxy", "action" : "CREATED", "asset" : { "id" : "31c950c8eeeab78336308177ae9c441c", "assetId" : "bnBtLXByb3h50jMxYzk1MGM4ZWV1YWI30DMzNjMwODE3N2F10WM0NDFj", "format" : "npm", "name" : "concrete" } }

⁸⁴⁶ <https://requestbin.com/>

Events share common fields, described in detail below:

Header	Description
nodeId	A UUID that identifies which Nexus Repository Manager node the event originated from
timestamp	The ISO 8601 representation of the time the event occurred
initiator	The userId or "anonymous" for system events

Table 19.2. Common Event Fields

Below, you will find examples of many Payloads that are returned, to help you get up and running with webhooks in Nexus Repository Manager.

Example Audit Payload

This is an example payload returned when a user is created inside of Nexus Repository Manager:

Global Audit Payload
<pre>{ "nodeId": "7FFA7361-6ED33978-36997BD4-47095CC4-331356BE", "initiator": "admin/127.0.0.1", "audit": { "domain": "security.user", "type": "created", "context": "testuser", "attributes": { "id": "testuser", "name": "test user", "email": "test@test.com", "source": "default", "status": "active", "roles": "nx-admin, nx-anonymous" } } }</pre>

Field	Description
audit:domain	A string that identifies the domain where the event occurred
audit:type	A string that identifies action type that occurred on the object
audit:context	A string that identifies the object the event refers to
attributes	A list that describes the attributes on the object that the event occurred on

Table 19.3. Audit Event Fields

Example Repository Payload

This is an example payload returned when a PyPi proxy repository(see page 441) is created inside of Nexus Repository Manager:

Global Repository Payload	
{ "timestamp": "2016-11-14T20:19:34.525+0000", "nodeId": "7FFA7361-6ED33978-36997BD4-47095CC4-331356BE", "initiator": "admin/127.0.0.1", "action": "CREATED", "repository": { "format": "pypi", "name": "pypi-proxy", "type": "proxy" } }	

Field	Description
action	A string that identifies the action performed on the repository
repository:format	A string that identifies the repository format type
repository:name	A string that identifies the repositories name
repository:type	A string that identifies the type of repository

Table 19.4. Repository Event Fields

Example Repository Asset Payload

This is an example payload returned when an asset is created inside of Nexus Repository Manager:

Repository Asset Payload	
{ "timestamp" : "2016-11-10T23:57:49.664+0000", "nodeId" : "52905B51-085CCABB-CEBBEAAD-16795588-FC927D93", "initiator" : "admin/127.0.0.1", "repositoryName" : "npm-proxy", "action" : "CREATED", "asset" : { "id" : "31c950c8eeeab78336308177ae9c441c", "assetId" : "bnBtLXByb3h50jMxYzk1MGM4ZWV1YWI30DMzNjMwODE3N2F10WM0NDFj", "format" : "npm", "name" : "concrete" } }	

Field	Description
repositoryName	A string that identifies the repository where the event occurred
action	A string that identifies the action performed on the asset
asset:id	A UUID that identifies the assets ID (to be used with Groovy scripting)
asset:assetId	A UUID that identifies the assets ID (to be used with REST API) NEW IN 3.17
asset:format	A string that identifies the repository format type
asset:name	A string that identifies the asset name

Table 19.5. Repository Asset Event Fields

Example Repository Component Payload

This is an example payload returned when a component is created inside of Nexus Repository Manager:

Repository Component Payload

```
{  
    "timestamp": "2016-11-14T19:32:13.515+0000",  
    "nodeId": "7FFA7361-6ED33978-36997BD4-47095CC4-331356BE",  
    "initiator": "anonymous/127.0.0.1",  
    "repositoryName": "npm-proxy",  
    "action": "CREATED",  
    "component": {  
        "id": "08909bf0c86cf6c9600aade89e1c5e25",  
        "componentId": "bnBtLXByb3h50jA40TA5mYwYzg2Y2Y2Yzk2MDBhYWR1ODl1MWM1ZTI1",  
        "format": "npm",  
        "name": "angular2",  
        "group": "types",  
        "version": "0.0.2"  
    }  
}
```

Field	Description
repositoryName	A string that identifies the repository where the event occurred
action	A string that identifies the action performed on the component
component:id	A UUID that identifies the assets ID (to be used with Groovy scripting)
component:componentId	A UUID that identifies the assets ID (to be used with REST API) NEW IN 3.17
component:format	A string that identifies the repository format type
component:name	A string that identifies the component name
component:group	A string that identifies the component group
component:version	A string that identifies the component version

Bundle Development

 Available in Nexus Repository OSS and Nexus Repository Pro

Nexus Repository Manager is built on top of the OSGi container [Apache Karaf](#)⁸⁴⁷. This supporting core infrastructure provides a foundation for these editions. The functionality is encapsulated in a number of OSGi bundles. Each edition is composed of a number of bundles, that provide the specific features.

Bundles can provide further functionality for the back-end such as support for new repository formats, specific behaviour for components, new tasks, and any other additional functionality as well as new user interface components and modifications. They can also group a number of these features together in one bundle.

This section provides a high level overview and information to begin developing your own bundles for the Nexus platform, and specifically the Nexus Repository Manager.

Knowledge of Apache Maven and Java are required for your bundle development efforts. OSGi-related knowledge is highly relevant and beneficial. Please ensure to consult the documentation for the relevant projects, when necessary.

If you work on any bundle development and require any help or assistance, please contact the development team via:

- the [users mailing list](#)⁸⁴⁸
- or email to nexus-feedback@sonatype.com⁸⁴⁹

❗ This documentation is not complete and Sonatype encourages you to provide feedback to help us answer any questions you might have and improve this section as needed.

Topics in this section:

- [Installing Bundles](#)(see page 525)
- [Bundle Development Overview](#)(see page 527)
- [Support for a New Repository Format](#)(see page 528)
- [Contributing Bundles](#)(see page 530)

Installing Bundles

In order to have your features from your bundle available as part of the repository manager, the bundle needs to be loaded by the OSGi container.

⁸⁴⁷ <http://karaf.apache.org/>

⁸⁴⁸ <https://groups.google.com/a/glists.sonatype.com/forum/?hl=en#!forum/nexus-users>

⁸⁴⁹ <mailto:nexus-feedback@sonatype.com>

The default build assembles multiple bundles into features that form the foundation of Nexus Repository Manager OSS and Nexus Repository Manager Pro. A number of these definitions can be found in the [assemblies⁸⁵⁰](#) module. The supported distributions are defined in the modules `nexus-oss-feature` and `nexus-pro-feature`, which are part of the internal code-base.

Installing Bundles Into 3.15.0 and Later

If the plugin bundle has been built as a KAR file you can just drop the file into `$install-dir/deploy` and it will be automatically unpacked and deployed. KAR files are like the old “plugin bundles” in Nexus Repo 2, they contain the plugin jar as well as its dependencies and a feature file that describes the plugin.

Assuming the plugin project uses `org.sonatype.nexus.plugins:nexus-plugins` as its parent then you can use the `buildKar` Maven profile when building the plugin to make a deployable KAR file.

```
mvn -PbuildKar clean install
```

Note that KAR file support was added in version 3.15.0 of Nexus Repo.

Reference: <https://github.com/sonatype/nexus-public/blob/release-3.15.0-01/plugins/pom.xml#L359>

Installing Bundles Into 3.14.0 and Earlier

if you're using a Nexus Repo 3.14.0 or earlier, or the plugin is not available in KAR format, use this procedure.

An installation of the repository manager defines the feature it loads in `$data-dir/etc/nexus.properties` and additional features can be declared to be loaded there. E.g. to add `my-custom-feature` to an Nexus Repository Manager OSS installation you can change to:

```
nexus-features=nexus-oss-feature,my-custom-feature
```

The feature `my-custom-feature` is a Maven project that includes the desired bundles as dependencies. Alternatively you can add a specific feature via Karaf commands.

Bundles can be loaded via the Karaf console. To enable the console, set `karaf.startLocalConsole` in `bin/nexus.vmoptions` to true. This allows you to access the Karaf console by pressing enter after starting the repository manager with the `run` option.

The `bundle:install` command can be used to load a bundle into the container.

For development usage, you can set the local Maven repository as the source for any bundle loading with e.g.

```
config:property-set -p org.ops4j.pax.url.mvn  
org.ops4j.pax.url.mvn.defaultRepositories "file:${user.home}/.m2/repository@id=system.repository@snapshots"
```

⁸⁵⁰ <https://github.com/sonatype/nexus-public/tree/master/assemblies/>

Once your bundle is installed, it will be displayed as part of the output from `bundle:list`. With the local Maven repository configured as a source, you can rebuild your bundle and get it reloaded and the repository completed restarted with:

```
bundle:update 270  
bundle:refresh  
system:shutdown -f -r
```

This process ensures that bundles are updated, imports are correctly picking up any changes and the full repository manager runs through the full start-up life-cycle.

Bundle Development Overview

The preferred way to write bundles is to use Java as the implementation language and Apache Maven as the build system. The [public code-base of Nexus Repository Manager OSS](#)⁸⁵¹ can be used as a starting point to investigate existing bundles and their source code. The easiest way to create a new bundle project is to replicate a bundle with a similar functionality. Inspect the source code of bundles with similar functionality, and read the JavaDoc documentation for the involved classes.

To gain access to all the components needed for your bundle development, you have to proxy the Sonatype grid repository with the URL:

```
https://repository.sonatype.org/content/groups/sonatype-public-grid/
```

Set up your project to include inheriting from the parent of all the Nexus Repository Manager OSS bundles with the version you are targeting as follows.

Inheriting from the nexus-plugins Parent

```
<parent>  
  <groupId>org.sonatype.nexus.plugins</groupId>  
  <artifactId>nexus-plugins</artifactId>  
  <version>3.0.0-SNAPSHOT</version>  
</parent>
```

⁸⁵¹ <https://github.com/sonatype/nexus-public>

- ❗ It is best to use the identical version of the parent as the Nexus Repository Manager instance on which you want to run your bundle. When developing a bundle you are using large parts of internals, which are subject to change from one version to another. This same logic applies to any dependencies as well.

A bundle Maven project creates a custom build output file in the form of an OSGi bundle. Enable this by changing the packaging to `bundle`. In addition, you need to add the `karaf-maven-plugin` and any needed dependencies. Inspect the `pom.xml` files for specific bundle in the `plugins` directory for further details.

These dependencies pull in a large number of transitive dependencies that expose Nexus Repository Manager functionality and other libraries to your project. Depending on the type of bundle and functionality you aim to create, additional dependencies and other details can be added to this minimal project setup. A large number of further classes is available and can be used as part of your bundle development.

With the exception of interfaces and code that is required to be accessible from modules outside of the format bundle, all code should be nested within an `internal` directory that will be isolated by the OSGi run-time container.

Once you have created your Maven project as described above, you can build the bundle with `mvn clean install`.

Support for a New Repository Format

This section examines the efforts required to implement support for a new repository format in the Nexus Repository Manager. By default, the repository manager includes support for various repository formats including `raw-format`, `maven2-format` and others.

When considering to implement support for a new repository format, it is important to get a good understanding of the format itself as well as the tools and the community working with the format. It might even be necessary to read and understand the source code of potential native, upstream implementations to support a format.

Following are a few questions that can provide some useful answers leading to a better understanding of the format

and necessary steps for implementation;

- What is this format all about?
- What tools (client/server) are involved?
- What communication is performed between client and server?
- Do any protocols or specifications exist?
- What authentication method needs to be supported by the repository manager?
- How can the repository manager authenticate against a proxied remote server?

- How does the concepts of components and assets used in Nexus Repository Manager map to the format?
- What is the best way to map the component identifier of name, version and group to the format?
- What format specific attributes should be stored as components and assets?
- Is it necessary to rewrite proxied metadata? E.g. proxied metadata contains absolute URLs to proxied server that it has to rewrite to point to repository manager.
- Are there any special features that should be considered?

To provide sufficient support for users, a new repository format needs to include a number of features:

- proxying components from remote repositories
- storing and managing components in a hosted repository
- exposing multiple repositories to users as a single repository group
- format-specific search criteria and the related search functionality

Depending on the specifics of the repository format being implemented a number of other features have to be provided or can optionally provide additional value to the user:

- any required tasks for maintenance of the repositories and their content
- client side tools to allow the standard tools to interact with the repositories on the repository manager
- custom features to display information about the repositories or their content in the user interface

The implementation of all these features for the raw -format can be found in the module `plugins/nexus-repository-raw`. The raw format is a good example code-base to expose as it presents the most simplistic repository format.

The Maven repository format as used by Apache Maven and many other tools is implemented in the `plugins/nexus-repository-maven` module. It can serve as another, slightly more complex, example. Examining the code base can be especially useful, if you know the Maven repository format.

Format, Recipe, and Facet

Extending Format allows you define support for your new repository format. Proxy, hosted and group functionality are implemented in a corresponding Recipe implementation each. The recipe enables the system to configures the view of a repository. It configures the facets that decorate the implementation, and matches up routes with appropriate handlers. Some handlers like the `SecurityHandler` are required for all repositories, while others are used to implement format specific functionality like managing the content (i.e. `RawContentHandler`).

Facets are used to decorate the format and provide additional functionality like proxy support (e.g. `RawProxyFacet`).

Each format plugin is required to extend `RepositoryFormatSecurityConfigurationResource` to provide security configuration. This simple implementation can be used to enhance the security rules as necessary.

Storage

An addressable component in a repository is described as a Component. Typically it defines common metadata like name and version and acts as the parent for one or multiple assets. An Asset represents binary content of any type - usually a JAR file, ZIP archive or some other binary format and additional files associated with the package (i.e. pom.xml for maven). Some metadata is automatically collected for Assets, like check-sums, while each format can also contribute its own specific metadata. An asset should always have a sha1 check-sum, but certain formats may require other types of check-sum and should extend the Asset.attributes.checksum map as required to store these.

User Interface

The user interface for supporting a new repository format is following a standard-pattern and is implemented as a recipe in the nexus-coreui-plugin bundle in src/main/resources/static/rapture/NX/coreui/view/repository/recipe/. These merely compose configuration for specific facets which should be implemented in .../repository/facet.

If a given format requires any additional specific configuration you have to add a new facet configuration screen with the required fields. They have to be mapped to the key/value map called attributes of the repository. E.g. a repository format foo has to be mapped to attributes.foo.someConfigProperty. New format configurations need to be registered in the views configuration of the controller in .../coreui/controller.Repositories.js.

Tasks

Tasks can be implemented for scheduled maintenance and similar tasks that operate on a repository as a whole. The Maven repository bundle includes a number of tasks that can serve as an example in the org.sonatype.nexus.repository.maven.tasks package.

Contributing Bundles

Ideally, any new bundles created yield significant benefits for the overall community of users. Sonatype encourages contribution of such bundles to the upstream repository and is offering support and help for such efforts.

The minimum steps for such contributions are:

- Create a pull request with the relevant changes to the [nexus-public repository](#)⁸⁵²
- Sign and submit a contributor license agreement to Sonatype

In further collaboration Sonatype will decide upon next steps on a case-by-case basis and work with you to:

⁸⁵² <https://github.com/sonatype/nexus-public>

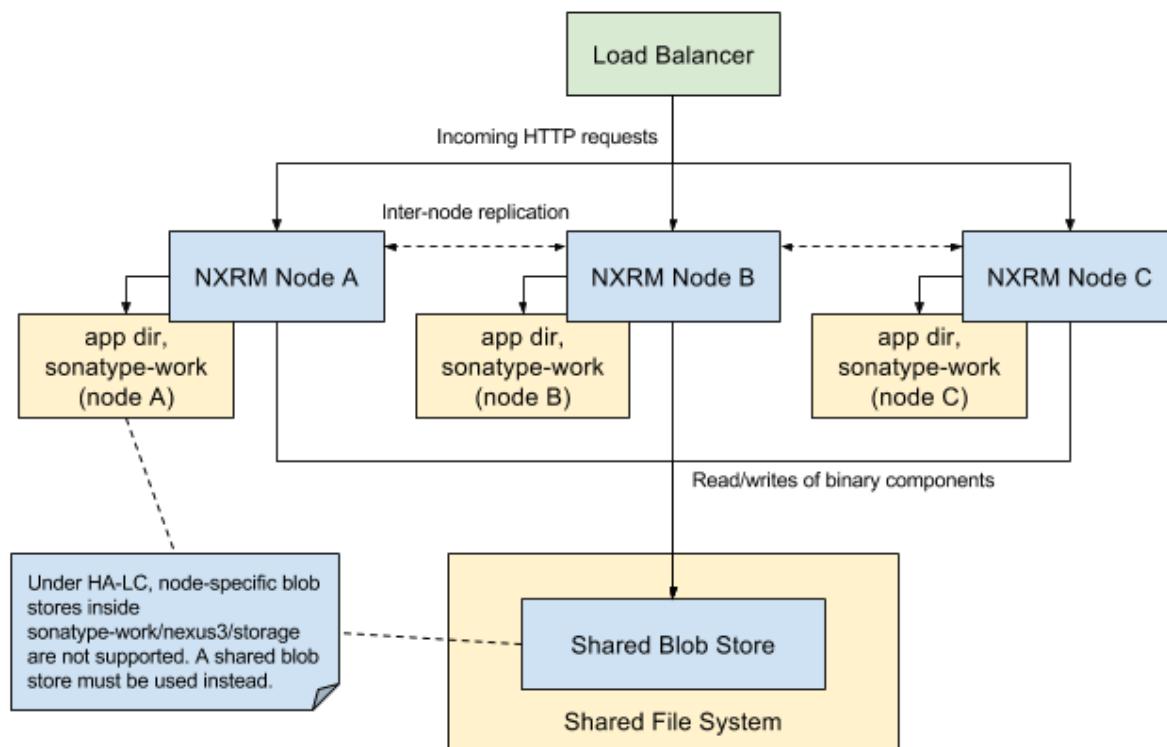
- Create sufficient tests
- Provide access to upstream repositories
- Facilitate other infrastructure such as CI server builds
- Help you with verification and testing
- Work with you on user documentation and outreach
- Expose your work to the user community

High Availability

 Available in Nexus Repository Pro only

Overview

High Availability Clustering (HA-C) is a feature designed to improve uptime by having a cluster of redundant NXRM "nodes" (instances) within a single data center. This feature allows you to maintain availability to your Nexus Repository Manager in the event one of the nodes becomes unavailable. A typical NXRM HA-C cluster is shown below:



Software and Hardware Requirements

Here's a list of the most important things you will need:

- A **test environment** to evaluate your cluster configuration and processes
- Nexus Repository Manager **Professional license** and **the latest version**
- A **load balancer**, such as [NGINX⁸⁵³](#) or [Apache HTTP⁸⁵⁴](#) or [AWS ELB⁸⁵⁵](#)
- **3 separate instances** of NXRM installed on **3 different systems** (e.g. 3 different EC2 instances) in a single datacenter
- **Network connectivity between the 3 different systems** so the NXRM can communicate with each other on several ports
- **Separate local and shared storage** (the difference will be described below)
- Each node needs to be configured according to our [general requirements](#)(see page 137).

In addition to these items, this guide will clarify the key concepts of **Node**, **Blob Store**, and **Hazelcast** and outline the operational considerations specific to HA-C.

Known Issues and Limitations

- The initial functionality intends to support a three-node cluster in a **single datacenter**. Cross-datacenter (WAN) replication is not supported at this time. For AWS, all nodes must be in the same availability zone.
- [Upgrading](#)(see page 563) an already established cluster to a new version of NXRM requires shutting down all nodes. Support for rolling upgrades is not yet available.
- Apt, Cocoapods, Conan, Conda, Go, Helm, p2 and R formats are currently disabled.

Topics in this section

- [Configuring Nodes](#)(see page 533)
- [Configuring Blob Stores](#)(see page 534)
- [Configuring Hazelcast](#)(see page 538)
- [Designing your Cluster Backup/Restore Process](#)(see page 543)
- [Initial Setup - High Availability](#)(see page 546)
- [Operating your cluster](#)(see page 550)
 - [JMX Lifecycle Operations](#)(see page 556)
 - [JMX Maintenance Operations](#)(see page 558)

⁸⁵³ <https://www.nginx.com/>

⁸⁵⁴ <https://httpd.apache.org/>

⁸⁵⁵ <https://aws.amazon.com/elasticloadbalancing/>

- [Upgrading your cluster](#)(see page 563)
- [Load Balancing](#)(see page 564)

Configuring Nodes

What is a Node?

We refer to each separate instance of NXRM in a high availability cluster as a node. Nodes can run on physical or virtual servers, containers, or cloud services like Amazon Web Services EC2 instances. While multiple nodes can exist on the same physical or virtual server, each bound to different ports for testing purposes, but such a setup is discouraged in production as it does not provide for a redundant clustered environment.

Each node needs its own local storage for the `$data-dir` folder. This folder must not be shared by any other node in the cluster. Within the local storage, a node will store:

- local instance configuration (network ports, `nexus.properties`, logging)
- local log files
- a complete copy of the internally managed [OrientDB](#)⁸⁵⁶ databases NXRM uses to store configuration and asset metadata
- a complete copy of the [ElasticSearch](#)⁸⁵⁷ indexes

In addition, all nodes need shared storage for [blob stores](#)(see page 534).

How many Nodes will you need?

NXRM HA-C is intended for use with 3 nodes. It will not work properly with less than 3.

Any node can accept a write, but for the write to be committed it must be replicated to a 'majority' of available running nodes. [OrientDB](#)⁸⁵⁸ calculates a majority using the following formula: (number of expected running nodes / 2) + 1

A write (e.g. publishing assets, changing system configuration) will fail if nodes in your cluster unexpectedly lose connectivity or die and fewer than the calculated majority are running. However you can still write to your NXRM HA-C environment with only 1 running node as long as the other members of the cluster have either a) not started yet or b) were shutdown cleanly. The NXRM HA-C feature includes [monitoring](#)⁸⁵⁹for this condition and controls to help you get your cluster running again.

⁸⁵⁶ <http://orientdb.com/orientdb/>

⁸⁵⁷ <https://www.elastic.co/products/elasticsearch>

⁸⁵⁸ <https://orientdb.com/docs/2.2/Distributed-Configuration.html#default-configuration>

⁸⁵⁹ <https://help.sonatype.com/display/NXRM3M/Operating+your+cluster#Operatingyourcluster-MonitoringNodeHealth>

- ⚠ Adding more than 3 nodes will result decreased performance due to the additional overhead incurred by database synchronization between the nodes. We recommend using no more and no less than 3 nodes in an HA cluster.

What data is replicated between Nodes?

NXRM HA-C uses [Hazelcast⁸⁶⁰](#) to keep the OrientDB databases completely in sync in an "Active-Active" configuration.

The ElasticSearch indices on the running nodes operate independently and do not share data; they do automatically index all content replicated in OrientDB.

Blobs, the assets (.jar, .xml, .tar.gz, docker images, etc.) in your repositories, are not replicated. They are stored in your blob stores.

- ⓘ Your next step is to plan your blob stores with [Configuring Blob Stores\(see page 534\)](#) before proceeding to [Initial Setup - High Availability\(see page 546\)](#).

Configuring Blob Stores

What is a Blob Store?

The binary assets you download via proxy repositories, or publish to hosted repositories, are stored in the blob store attached to those repositories. In traditional, single node NXRM deployments, blob stores are typically associated with a local filesystem directory, usually within the `sonatype-work` directory.

For NXRM HA-C however, the blob store location must be:

1. outside of the `sonatype-work` directory
2. read-write accessible by all nodes

- ⓘ Caution: Sonatype has very specific guidelines on choosing an appropriate filesystem for blob stores in a NXRM HA-C environment.

⁸⁶⁰ <https://hazelcast.org/>

Choosing a Type for your Blob Stores

NXRM HA-C stores blobs in shared storage, which can be either a shared file system or a cloud object store. Here are some recommendations to consider when choosing where to store your blobs. In general, file system blob stores may have better performance, but object store blob stores offer unbounded storage and convenience. Cloud object stores perform better if you are also running NXRM in the cloud with the same cloud provider.

- ❗ Do not copy or share the `sonatype-work` directory used by your Nexus Repository Manager nodes. Each node must be allowed to initialize its own private `sonatype-work` directory; data generated on first run is unique to each instance.

Your shared file systems, if File based, must exist outside of the `sonatype-work` directory.

Recommended Shared File Systems

NFS v4 is the recommended file system. On [Amazon Web Services](#)⁸⁶¹ (AWS), [Elastic Filesystem](#)⁸⁶² (EFS) can be used as an NFS v4 server. Whatever the file system type, the blob store location must be outside of the `sonatype-work` directory and read/write accessible by all nodes.

NFS v4

The recommended settings for an NFS v4 server in `/etc/exports`:

```
/srv/blobstores 10.0.0.0/8(rw,no_subtree_check)
```

The recommended settings for mounting the NFS filesystem on each node:

```
defaults,noatime,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,vers=4.1
```

- ⚠ Versions of NFS older than v4 are not supported.

⁸⁶¹ <https://aws.amazon.com/>

⁸⁶² <https://aws.amazon.com/efs/>

AWS Elastic Filesystem (EFS)

EFS acts as a limitless NFS v4 server and is an option if NXRM is running in AWS. Clients should mount the EFS volume with the same settings as NFS v4. Note that EFS performance will be lower than a dedicated NFS server.

Cloud Object Stores

Cloud object stores store your blobs using AWS S3. No shared file system is required if cloud object stores are used exclusively.

AWS Simple Storage Service (S3)

Blob stores can be configured to use [AWS Simple Storage Service](#)⁸⁶³.

Blobs are stored in an S3 bucket by using AWS REST APIs over HTTP. Object PUTs use multi-part uploads of approximately 5MB chunks. Since HTTP traffic to S3 blobstores introduces more I/O latency across the network, **it is recommended to use S3 only if NXRM is running on EC2 instances within AWS**

When setting up a bucket for the S3 blob store:

- NXRM will automatically apply a lifecycle rule to expire deleted content.
- The bucket can use server-side encryption with KMS key management transparently. Other methods of server side encryption are not supported.

The AWS user for accessing the S3 Blobstore bucket needs to be granted permission for these actions:

- s3:PutObject
- s3:GetObject
- s3:DeleteObject
- s3>ListBucket
- s3:GetLifecycleConfiguration
- s3:PutLifecycleConfiguration
- s3:PutObjectTagging
- s3:GetObjectTagging
- s3:DeleteObjectTagging
- s3>DeleteBucket
- s3>CreateBucket
- s3:GetBucketAcl ([used for problem diagnosis](#)⁸⁶⁴) **NEW IN 3.19.0**

Sample minimal policy where <user-arn> is the ARN of the AWS user and <s3-bucket-name> the S3 bucket name:

⁸⁶³ <https://aws.amazon.com/s3/>

⁸⁶⁴ <https://issues.sonatype.org/browse/NEXUS-19494>

```
{  
    "Version": "2012-10-17",  
    "Id": "NexusS3BlobStorePolicy",  
    "Statement": [  
        {  
            "Sid": "NexusS3BlobStoreAccess",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "<user-arn>"  
            },  
            "Action": [  
                "s3:PutObject",  
                "s3:GetObject",  
                "s3:DeleteObject",  
                "s3>ListBucket",  
                "s3:GetLifecycleConfiguration",  
                "s3:PutLifecycleConfiguration",  
                "s3:PutObjectTagging",  
                "s3:GetObjectTagging",  
                "s3:DeleteObjectTagging",  
                "s3>DeleteBucket",  
                "s3>CreateBucket",  
                "s3:GetBucketAcl"  
            ],  
            "Resource": [  
                "arn:aws:s3:::<s3-bucket-name>",  
                "arn:aws:s3:::<s3-bucket-name>/*"  
            ]  
        }  
    ]  
}
```

Access Denied Errors

Access denied errors are an indication that the user configured to connect to the bucket has insufficient permissions. AWS does not provide information specific to which permission is required to perform the failed action. For this reason NXRM will generically report access denied errors.

File Systems to avoid

Using the File Blob Store with NXRM HA-C relies on POSIX semantics. File systems known to be unreliable are:

1. glusterfs
2. FUSE based user space filesystems

Determining a Backup Solution

Backing up your blob store file systems is covered in detail in the [Designing your Cluster Backup/Restore Process](#)(see page 543) section.

- ⓘ Your next step is to review your network configuration with [Configuring Hazelcast](#)(see page 538).

Configuring Hazelcast

Network Preparation

The nodes in a Nexus Repository Manager cluster need to be able to communicate with each other over TCP/IP. For cluster communications, Nexus Repository Manager will use a range of ports starting with 5701. Each node in the Nexus Repository Manager cluster will require that an extra port is available; the extra ports used by Nexus Repository Manager will be bound sequentially by default.

For example, in a three-node cluster, each of the nodes in the cluster will need to have ingress opened on ports

5701

, 5702, and 5703. The nodes will use ephemeral ports, randomly selected by the operating system, for outbound (egress) communications. If outbound or inbound network communications need to be customized and may be blocked by a firewall or other network appliance, the ports used for cluster communications can be customized in the Hazelcast configuration.

- ⓘ For more information on customizing Hazelcast and the ports it uses, please see the documentation for [Hazelcast cluster configuration](#)⁸⁶⁵.

The nodes in a Nexus Repository Manager cluster will also replicate database transactions among the nodes in the cluster. The database requires ports 2424 and 2434 be open for ingress and egress to each of the rest of the nodes in the cluster.

⁸⁶⁵ <http://docs.hazelcast.org/docs/3.6/manual/html-single/index.html#setting-up-clusters>

Node Discovery

Hazelcast has multiple methods for discovering other nodes. In the default configuration, Nexus Repository Manager uses multicast to discover other Nexus Repository Manager nodes. This is done to simplify cluster configuration. However, multicast may not work reliably in all network environments.

To customize discovery, copy NEXUS_HOME/etc/fabric/hazelcast-network-default.xml to \$data-dir/etc/fabric/hazelcast-network.xml and adjust the settings enclosed in the <join> tag.

 In NXRM 3.6.1 or earlier, these changes must be applied directly to NEXUS_HOME/etc/fabric/hazelcast.xml, where NEXUS_HOME is where Nexus Repository Manager is installed.

Multicast Discovery

Multicast is the default discovery method and is recommended unless it is not supported on your network. To test multicast connectivity between nodes, we recommend using [iPerf](#)⁸⁶⁶. iPerf is freely available for Windows, Linux and Mac OSX under the BSD licence.

To test multicast connectivity between two nodes, each node will need to have iPerf installed (note: iPerf3 does not support multicast client testing). During testing, one node will act as the "server" while the other node acts as the "client." This is important during testing, and further, we suggest that each node be tested as a server and as a client to ensure proper two-way multicast communication.

Testing multicast from the server side

```
iperf -s -B 224.2.2.3 -p 54327 -i 1
```

Testing multicast from the client side

```
iperf -c 224.2.2.3 -p 54327 -u -i 1
```

The address and port 224.2.2.3:54327 was chosen because this is the default port and address that will be used by Nexus Repository Manager during the node discovery. When the client is able to successfully connect and communicate with the server node, the client will output a set of brief diagnostic messages indicating how much data was sent and received. The following is a sample of the output from a client in a successful session:

Sample Output

⁸⁶⁶ <https://iperf.fr/>

```
Client connecting to 224.2.2.3, UDP port 54327
Sending 1470 byte datagrams
Setting multicast TTL to 1
UDP buffer size: 208 KByte (default)

[ 3] local 10.10.0.102 port 33743 connected with 224.2.2.3 port 54327
[ ID] Interval      Transfer     Bandwidth
[ 3]  0.0- 1.0 sec   129 KBytes  1.06 Mbits/sec
[ 3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[ 3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[ 3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[ 3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[ 3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[ 3]  6.0- 7.0 sec   129 KBytes  1.06 Mbits/sec
[ 3]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[ 3]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[ 3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[ 3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 3] Sent 893 datagrams
```

The advantage of the using multicast for Nexus Repository Manager node discovery is that nodes can be added and removed from the cluster without cluster administrators needing to perform configuration or configuration changes. However, routers may not be able to route multicast requests properly between subnets, or multicast may be disabled altogether. In these situations the cluster configuration can be done manually. If multicast is not available, either [AWS Discovery](#)(see page 540) (if you are running NXRM inside AWS on EC2 instances) or [TCP/IP Discovery](#)(see page 542) can be used.

For a production instance, we advise that the default address and port (224.2.2.3:54327) is not used, to avoid another cluster within the same network and using the same default configuration from unintentionally joining an existing cluster.

AWS Discovery

Nexus Repository Manager can be deployed on cloud-computing services, such as Amazon Web Services (AWS), where multicast discovery is not supported. Hazelcast AWS discovery is recommended.

AWS Environment

The NXRM servers need permissions to find other nodes, granted through [AWS Identity and Access Management](#)⁸⁶⁷ (IAM). The following configuration settings should be used:

1. The EC2 instances running NXRM should be assigned an `InstanceProfile`
2. The `InstanceProfile` should have an `InstanceRole` with a policy granting the permission `ec2:DescribeInstances` on all resources.

In CloudFormation, the role would look similar to this:

⁸⁶⁷ <https://aws.amazon.com/iam/>

```
"InstanceRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ec2.amazonaws.com"
                    },
                    "Action": [
                        "sts:AssumeRole"
                    ]
                }
            ],
            "Policies": [
                {
                    "PolicyName": "hazelcastDiscovery",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "ec2:DescribeInstances"
                                ],
                                "Resource": ["*"]
                            }
                        ]
                    }
                }
            ]
        }
    }
}
```

NXRM Configuration for AWS Discovery

To configure Hazelcast for automatic node discovery, you need the IAM role name, AWS region, and security group of the EC2 instances. Find the `<join>` tag in `$install-dir/etc/fabric/hazelcast-network.xml`. Then, edit the file for each node:

1. Change the value in `<multicast enabled="true">` to "false".
2. Update the `<discovery-strategies>` section as explained below.
3. Save the file.
4. Reboot each node in the cluster.

The `$data-dir/etc/fabric/hazelcast-network.xml` file with the modified properties will look similar to this:

```
<join>
    <!-- deactivating other discoveries -->
    <multicast enabled="false"/>
    <tcp-ip enabled="false"/>
    <aws enabled="false"/>

    <discovery-strategies>
        <discovery-strategy enabled="true" class="com.hazelcast.aws.AwsDiscoveryStrategy">
            <properties>
                <!--
                    | Required: use the command 'aws iam list-instance-profiles' on the EC2 instance to locate the
                    name
                    |
                    | of the role used by the IAM Instance Profile you created previously
                    --
                <property name="iam-role">EC2_IAM_ROLE_NAME</property>

                <!-- Required: set this to the region where your EC2 instances running Nexus Repository Manager
                    are located -->
                <property name="region">us-west-1</property>

                <!--
                    | The next few options give you a choice to how the nodes are enumerated. You can specify:
                    | * Just a security group name, if the security group only contains Nexus Repository Manager
                    hosts
                    |
                    | * or a tag-key/tag-value pair, if you have tagged the EC2 instances
                    | * or a combination of the two, if the security group has other hosts in it.
                    --
                <property name="security-group-name">EC2_SECURITY_GROUP_NAME</property>

                <!-- example tag idea, you are free to use whatever tag naming convention you need-->
                <property name="tag-key">Purpose</property>
                <property name="tag-value">Nexus Repository Manager</property>

            </properties>
        </discovery-strategy>
    </discovery-strategies>
</join>
```

- ⓘ For more information on configuring Hazelcast in an AWS environment, please see the documentation for the [AWS EC2 discovery plugin for hazelcast](#)⁸⁶⁸.

TCP/IP Discovery

When multicast discovery and AWS discovery are not available, TCP/IP discovery can be configured. TCP/IP discovery requires the IP address of each node to be included in the configuration. Please see the

⁸⁶⁸ <https://github.com/hazelcast/hazelcast-aws>

documentation for [Hazelcast cluster configuration](#)⁸⁶⁹. Find the <join> tag in \$install-dir/etc/fabric/hazelcast-network.xml. Then, edit the file for each node:

1. Add or set the following in \$data-dir/etc/nexus.properties: nexus.hazelcast.discovery.isEnabled=false
2. Update the <join> section as explained below.
3. Save the file.
4. Reboot each node in the cluster.

In this case the \$data-dir/etc/fabric/hazelcast-network.xml file with the modified properties will look similar to this:

```
<join>
  <multicast enabled="false"/>
  <tcp-ip enabled="true">
    <member-list>
      <member>10.0.1.10</member>
      <member>10.0.1.11</member>
      <member>10.0.1.12</member>
    </member-list>
  </tcp-ip>
  <aws enabled="false"/>
</join>
```

Designing your Cluster Backup/Restore Process

You must have a reliable, tested process for backing up your Nexus Repository Manager deployment .

This section will cover what you need to create a backup and restore process appropriate for your High Availability deployment .

High-level Overview

An appropriate backup process for Nexus Repository Manager needs to capture state from two separate, but interdependent parts of your deployment:

- the contents of each blob store
- a copy of the local storage for each node (i.e. the \$data-dir directory, including the OrientDB databases within)

In the worst case scenario, a Nexus Repository Manager High Availability environment can be recreated minimally from:

⁸⁶⁹ <http://docs.hazelcast.org/docs/3.6/manual/html-single/index.html#setting-up-clusters>

- the contents of each blob store
- a complete set of OrientDB database exports from one of the nodes participating in the cluster

Backing up Shared File Systems for your Blob Store(s)

 Nexus Repository Manager requires that you choose and execute a separate backup process appropriate for the file systems under your blob store(s).

Generic NFS

Tools that back up the file system under the blob store can be safely executed while Nexus Repository Manager is running.

NFS over Amazon Elastic File System (EFS)

If you are using file backed blob stores on top of Amazon Elastic File System, you will need to review [Amazon's documentation specific to backing up EFS](#)⁸⁷⁰.

Amazon Simple Storage Service (S3)

The backup process for your S3 backed blob stores will involve creating a separate S3 "bucket" and periodically synchronizing the content from the source bucket under your blob store(s).

The [AWS Command Line Interface \(CLI\)](#)⁸⁷¹ provides an `s3 sync` command that you can invoke periodically to perform this:

- <http://docs.aws.amazon.com/cli/latest/reference/s3/sync.html>

There are also a number of third party tools that can perform this task.

Backing up local storage for your Nodes

Each node in your Nexus Repository Manager High Availability cluster has its own separate `$data-dir` directory. The `$data-dir /db` directory within contains OrientDB databases and requires special treatment.

The `db` directory cannot be backed up by basic file system backup tools while Nexus Repository Manager is running. Instead create an *Admin - Export databases for backup* task as described in the [Backup and Restore](#)(see page 294) section.

Please note:

⁸⁷⁰ <http://docs.aws.amazon.com/efs/latest/ug/efs-backup.html>

⁸⁷¹ <https://aws.amazon.com/cli/>

- All nodes in your Nexus Repository Manager cluster will automatically enter Read-Only mode when the backup task is running on any of the nodes
- It is recommended to schedule the backup task on at least 2 nodes. The tasks should run at different times to avoid concurrency issues. The backup from any node can be used to perform a restore, but running on more than one node ensures that there isn't a single point of failure for the backup.
- The folder specified in the backup task for a node must be available and writable by the user running Nexus Repository Manager on the nodes
- The backup task produces a set of files each time it is run that must be protected. Be sure to use a file system backup tool to store these files separately.

For the rest of the content within the `$data-dir` directory, traditional file system backup tools will suffice. It is safe to skip backing up the following directories within `$data-dir`:

- - `backup`
 - `cache`
 - - `db`
(backed up via the *Admin - Export databases for backup* task)
 - `elasticsearch` (generated via the *Repair - Rebuild repository search* task)
 - `logs` (see the [Operating your cluster](#)(see page 550) section for preserving logs)
 - `tmp`

 We recommend you execute backup of your local storage at the same starting time as your blob store file system backup.

Restoring from backup

 If you determine you need to restore your Nexus Repository Manager deployment from backup, you must first focus only on starting one node and stabilizing it before you add new nodes to the High Availability cluster.

Start this process by ensuring all Nexus Repository Manager nodes in your cluster are offline.

Assuming all nodes in your cluster are offline:

1. Identify the host that will be the first node of your restored deployment. Locate the Orient DB exports from that node and identify the date and time they were taken. Use this same (or as close to) date and time when restoring the blob store file systems.
2. Restore the blob store file systems and host connectivity. Be sure to restore the blob store file system to the exact same absolute file system path used previously; don't try to restore to a different file system path.
3. Restore the local storage (`$data-dir`) for the node. Again, preserve the same absolute file system paths used previously.
4. Remove the following directories from `$data-dir/db`
 - component
 - config
 - security
5. Copy the complete set of OrientDB exports from that node to `$data-dir/restore-from-backup` for restoration (**Note:** For version 3.10.0 or earlier use `$data-dir/backup` as the restore location).
6. Start Nexus Repository Manager on only this node.

Confirm that your instance is in a stable state before proceeding:

- Inspect configuration and confirm it matches expectations
- Attempt to retrieve and/or publish new components

Once the first node is online and stable, it is safe to proceed in bringing additional nodes back into your cluster. Focus on adding only one node at a time to the cluster. On each of the additional nodes, delete the following folders from within `$data-dir /db` prior to starting Nexus Repository Manager

- component
- config
- security

The OrientDB databases removed from the additional node will automatically be rebuilt from the instances already running in the High Availability cluster.

Initial Setup - High Availability

This section will cover the steps for enabling High Availability in your Nexus Repository Manager deployment.

⚠ This document assumes that you have completely read and prepared your choices for [Configuring Blob Stores](#)(see page 534), [Configuring Hazelcast](#)(see page 538), and [Designing your Cluster Backup/Restore Process](#)(see page 543).

Do not proceed until you have a complete plan ready for all 3.

Once you begin, focus on stabilizing one single node at a time.

Node Deployment Steps for High Availability

First Node

1. Follow the usual [Installation Methods](#)(see page 170). Create the file `sonatype-work/nexus3/etc/nexus.properties` with the following contents:

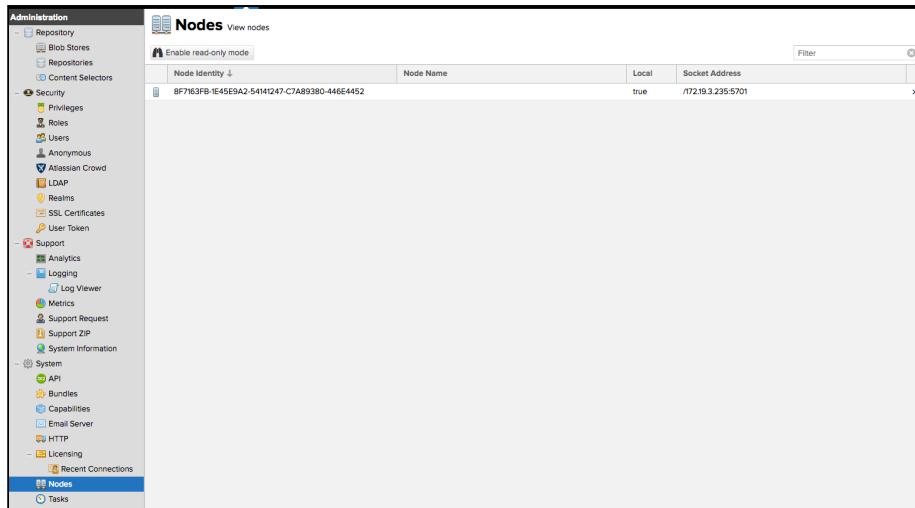
```
# Jetty section
application-port=8081
application-host=0.0.0.0
nexus-args=${jetty.etc}/jetty.xml,${jetty.etc}/jetty-http.xml,${jetty.etc}/jetty-requestlog.xml
nexus-context-path=/


# Nexus section
nexus-edition=nexus-pro-edition
nexus-features=\n    nexus-pro-feature

nexus.clustered=true
# nexus.licenseFile is only necessary for the first run
# replace /path/to/your/sonatype-license.lic with the path to your license, and ensure the user
# running Nexus Repository manager can read it
nexus.licenseFile=/path/to/your/sonatype-license.lic
```

2. Next, deploy your chosen [hazelcast configuration](#)(see page 538) to the path `sonatype-work/nexus3/etc/fabric/hazelcast-network.xml`. For NXRM 3.6.1 or earlier, these changes must be made in `NEXUS_HOME/etc/fabric/hazelcast.xml`, where `NEXUS_HOME` is where Nexus Repository Manager is installed.

3. Start Nexus Repository Manager. Log in as an administrator, then visit the *System → Nodes* screen. You should see your first node, like this example:



The screenshot shows the 'Nodes' screen in the Nexus Repository Manager interface. On the left is a navigation sidebar with sections like 'Administration', 'Security', 'Support', 'System', and 'Nodes'. The main area is titled 'Nodes View nodes' and contains a table with one row. The table has columns for 'Node Identity' (containing the unique ID 'BF7163FB-1E45E9A2-54141247-C7A89380-446E4452'), 'Node Name' (empty), 'Local' (set to 'true'), and 'Socket Address' ('/72.19.3.235:5701'). There is also a 'Filter' input field and a 'Delete' button for the selected node.

You may want to assign a friendly *Node Name* to each node in your cluster to make it easier to identify. Without a node name, screens that refer to individual nodes will display the unique ID labeled *Node Identity* in the screenshot above. Simply click on the entry in the list and you will have the ability to set a node name.

When Nexus Repository Manager is started with `nexus.clustered=true` and a PRO license, it will not create the default blobstore or initial example repositories. You are free to set these up now, or later as you add nodes.

Second Node and Beyond

! Do not copy the `sonatype-work` directory to your additional nodes. Each node must be allowed to initialize its own private `sonatype-work` directory; data generated on first run is unique to each instance.

Starting a second node off of a copy of the `sonatype-work` directory will result in the inability to correctly form a cluster.

Again follow the usual [Installation Methods](#)(see page 170). Before starting Nexus Repository Manager, repeat the steps you performed on the first node:

1. Create the file `sonatype-work/nexus3/etc/nexus.properties` with the same contents as the first node (note that you will have to choose a different port number only if you're running two or more nodes on a single host):

```
# Jetty section
application-port=8081
application-host=0.0.0.0
nexus-args=${jetty.etc}/jetty.xml,${jetty.etc}/jetty-http.xml,${jetty.etc}/jetty-requestlog.xml
nexus-context-path=/

# Nexus section
nexus-edition=nexus-pro-edition
nexus-features=\nexus-pro-feature

nexus.clustered=true
# nexus.licenseFile is only necessary for the first run
# replace /path/to/your/sonatype-license.lic with the path to your license, and ensure the user
running Nexus Repository manager can read it
nexus.licenseFile=/path/to/your/sonatype-license.lic
```

2. Deploy your chosen hazelcast configuration to the path sonatype-work/nexus3/etc/fabric/hazelcast-network.xml
3. Start Nexus Repository Manager

After each node joins the cluster, confirm it is visible in the *System → Nodes* screen. Set a node name for each node as desired. Repeat this section for each node you wish to join your High Availability Cluster until all are running.

Enabling High Availability on an existing Nexus Repository Manager Deployment

If you already have a single node Nexus Repository Manager deployment, you will still need to read and prepare your choices for [Configuring Blob Stores](#)(see page 534), [Configuring Hazelcast](#)(see page 538), and [Designing your Cluster Backup/Restore Process](#)(see page 543). You will also need to upgrade your single node deployment to a version of Nexus Repository Manager that supports [High Availability](#)(see page 531) before you attempt to establish a cluster. All nodes within a cluster must be running the exact same version of Nexus Repository Manager.

You additionally may have to design a strategy for migrating the content on your existing blob stores to a shared filesystem accessible to all nodes in the cluster. Moving blobs from one blobstore to another can be achieved using Dynamic Storage.

Blob Store Content Migration

All blob stores used by HA-C nodes must be shared. If you need to move blobstore blobs to new blobstores in preparation for sharing between nodes, first perform that procedure in a single node configuration using the [Dynamic Storage](#)⁸⁷²operations.

Configuration

Once you have all the pieces in place, enable high availability by performing the steps listed in the Second Node and Beyond section above.

Operating your cluster

⚠ It is recommended that the nodes are started in the reverse order in which they were shutdown wherever possible (for example, if they were shutdown in the order A, B, C, then C should be started first, followed by B and then A). The last node shutdown will have the most current data and should be the first one started up after a full cluster shutdown.

Startup and Confirming Node Connectivity

After enabling HA-C for your Nexus Repository Manager installation, check the console or log file after launching a clustered node to confirm that all nodes have been detected.

When you start the nodes, you will see a message in the nexus.log confirming the connection of the cluster members, like the one below:

```
2017-11-06 11:04:12,045-0500 INFO [hz.nexus.generic-operation.thread-0] *SYSTEM
com.hazelcast.internal.cluster.ClusterService - [172.19.3.78]:5703 [nexus] [3.7.8]

Members [3] {
    Member [172.19.3.78]:5701 - 6f4df1bc-606d-4821-8a3b-0980f093abd1
    Member [172.19.3.78]:5702 - 436158e4-2865-40fe-bfaa-d350db8dedb1
    Member [172.19.3.78]:5703 - 7ff93c39-23b3-4bf5-9b8d-51264cceceee2 this
}
```

⁸⁷² <https://guides.sonatype.com/repo3/technical-guides/scaling-dynamic-storage/>

Running in Docker

Running your Nexus Repository Manager cluster within a container network is supported. However, there is an important consideration. When running Nexus Repository Manager inside of a docker container, the default behavior of the "docker stop" command is to first send a TERM signal, but if the process has not quit within 10 seconds, a KILL signal is sent. This has been known to cause Nexus Repository Manager clusters to become corrupted.

 **WARNING**

When running Nexus Repository Manager nodes inside docker containers, you will need to specify an extended timeout, such as

```
docker stop --time=120 <CONTAINER_NAME>
```

to allow nodes to exit the cluster cleanly.

Shutdown

A node leaves the cluster through a clean shutdown of Nexus Repository Manager. A proper shutdown is performed by using the `nexus.exe stop` command on Windows-based systems and `nexus stop` on all others. When the node is leaving by a clean shutdown, the node is unregistered from the cluster. If a node that is registered in the cluster becomes abruptly unavailable (for example, the node's network link is broken, or the node's operating system crashes), that node may remain registered in the cluster despite no longer participating in database replication.

 The clean shutdown is an important consideration when designing scripts for stopping a Nexus Repository Manager instance.

Nexus Repository Manager will initiate a shutdown if it receives the TERM signal, which is the default signal sent with the kill command on unix-like operating systems. A clean shutdown is also important for recovering from an outage event.

Backup

Make sure to take some time to read the section on [designing your backup and restore plan](#)(see page 543). Running Nexus Repository Manager in a cluster does not reduce the need for a robust backup strategy. With a cluster, you will need to specify the repository manager instance where the task will execute. This is

necessary because the backup will result in exported database backup files. These files will be placed in the configured location, on the chosen node's file system. However, since the task is relegated to a single node, this introduces the possibility that the database backup may not be executed if the node is no longer participating in the cluster. In some environments it may be preferable to configure two different nodes to execute backup tasks on alternating schedules. While the [Backup and Restore\(see page 294\)](#) task is running, the Nexus Repository Manager cluster will be in read-only mode.

Testing your Backup/Restore process

We encourage you to test your [restore process\(see page 545\)](#) for disaster recovery minimally every 6 months. We also strongly suggest that you test the disaster recovery plan prior to upgrading Nexus Repository Manager to a new release.

 Please keep in mind Nexus Repository Manager requires that you choose and execute a separate backup process of your blob store(s) appropriate for the file system.

Node Names

When clustered, Nexus Repository Manager offers the ability to specify an alias for each node. Behind the scenes, Nexus Repository Manager generates a unique identifier for each node that joins a cluster. The alias that is specified by the administrator allows a logical mapping between the system-generated identifier and the specified name. The alias for a node is stored in `sonatype-work/nexus3/etc/node-name.properties`. The name can be updated in the Nexus Repository Manager user interface or directly in the properties file. If the name is updated directly in the properties file, it is necessary to restart the renamed node for the change to take effect. When an alias has been provided for a node, that alias will be used in place of the identifier where appropriate. For instance, the alias will be used in logging messages and in the user interface.

Maintaining log files

Clustered Nexus Repository Manager nodes will create the same set of log files as unclustered installations. However, in a cluster, the name of the node will be added to each log message. The name that is used in the log messages will be the given alias or default to the node's system-generated identifier. The log files on each node will be rotated daily. The log files for the previous day will be renamed and compressed to save disk space.

The Nexus Repository Manager user interface contains a log viewer available to administrators. This log viewer is limited to only display log messages that have been emitted on the local node. In an environment where traffic is routed through a load balancer, this view may not be helpful. In that case, we suggest that a log aggregation system be put in place. Having a node identifier as a part of each log entry should enable better searching within such a system.

Verifying Synchronization

At run-time, the repository manager user interface allows you to view the status of the clustered nodes.

See [Nodes \(see page 263\)](#) for details on viewing active nodes in a cluster.

-  In the event a single node loses connection to the cluster, the remaining nodes will continue to make decisions on which data changes are valid. The disconnected node will reject further writes until it rejoins the cluster.

Database Quorum and Cluster Reset

To maintain data consistency, Nexus Repository Manager forces database transactions to be accepted and written to multiple nodes in the cluster. The goal is that all transactions are replicated within the cluster before they are accepted. However, this introduces some situations that need explanation. To allow for continued operation, even during some failures, the transactions are not expected to replicate to all nodes in a cluster before the transaction is accepted. The Nexus Repository Manager cluster does expect that the transaction is replicated to the majority of nodes within a cluster. This means that in a cluster with three nodes, no less than two nodes must accept a write before the transaction is considered a success. To calculate the size of the majority, Nexus Repository Manager will use the following equation:

 $\text{ROUNDED_DOWN}(\ $N / 2) + 1$

Where $\$N$ represents the number of nodes registered in the cluster and `ROUNDED_DOWN` finds the largest whole number less than or equal to the given parameter.

Nodes register with the cluster when they are launched.

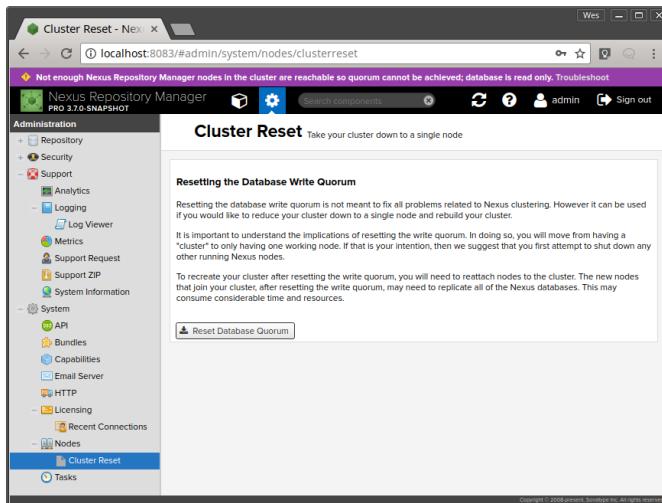
Although Nexus Repository Manager clustering replicates data across all nodes in the cluster, a cluster can be reduced to a single node. A single node cluster will accept write transactions. You can still write to your Nexus Repository Manager cluster environment with only one running node as long as the other members of the cluster have either been shutdown cleanly or they have not started yet. If nodes in your cluster unexpectedly lose connectivity or die and fewer than the calculated majority are running, then attempts to write (such as publishing assets, changing system configuration) will fail.

Because of the quorum calculation and cluster registration algorithms, it is possible that Nexus Repository Manager's cluster table can enter a state where it does not accurately reflect the true state of your cluster. For example, if a registered member is abruptly removed from the cluster and a fresh installation is added to replace the failed node, then Repository Manager clustering may be calculating the write quorum for database transactions still considering the removed node. In this example, Nexus Repository Manager will

continue to operate. However, after each abrupt removal of a node, it becomes more likely that a cluster could have difficulty achieving write quorum.

Troubleshooting

If your cluster enters a state where write quorum cannot be achieved, then Nexus Repository Manager will present the option to reset your cluster. The write quorum not reached warning appears as a horizontal bar across the top of the Nexus Repository Manager user interface, as shown in the screen capture below:



Before resetting your Nexus Repository Manager cluster, you should attempt to manually resolve the cluster issues. There are a few ways you can manually resolve the issues.

- Try to cleanly shutdown all clustered nodes, except one, and then try to bring your cluster back online by restarting the nodes one at a time waiting for each to finish starting before starting the next.
- You can also try to reconnect or restart nodes that may have been killed or disconnected. If the restarted nodes are not meant to be permanent members of the cluster, then performing a clean shutdown after restarting or reconnecting will properly remove the nodes from the cluster.

As a last resort, you can click the *Troubleshoot* link in the write quorum warning. On the cluster reset page, you will see the *Reset Database Quorum* button. When you click on the button, all entries are removed from the cluster table except for the node that processes the request to reset the cluster. In practice, this means that all other nodes need to be shutdown and removed from the load balancer rotation before resetting the cluster.

WARNING

It is only safe to reset the cluster once all other nodes have been cleanly shutdown.

After the reset completes, the cluster will immediately begin accepting writes on the remaining node. Nodes can then be added to the cluster and placed back into the load balancer rotation.

Monitoring Node Health

Once your HA environment is set up, you should monitor the health of the nodes in your cluster. The read-only status of the cluster is visible at the `http://<serveripaddress>:<port>/service/metrics/data` endpoint, as "readonly.enabled", under "gauges". See the [support article⁸⁷³](#) for the HTTP endpoints that HA-C exposes and/or the [support article⁸⁷⁴](#) for enabling JMX, for more information.

You can also use the load balancer friendly [status resource⁸⁷⁵](#) to check the health of the nodes in an HA cluster.

Task Management

In a Nexus Repository Manager cluster, it is necessary to consider which node or nodes execute a task. Knowing how the tasks run in a cluster will be an important element of monitoring cluster health. Nexus Repository Manager now places task logging output in a separate file that corresponds to the execution of the task. Many tasks will be executed on a random single node in the cluster to balance the resource usage across the cluster. Other tasks may execute simultaneously on all nodes in the cluster. Monitoring of the task must take into account the behavior of the execution of the task. The node requirements for common tasks are listed below.

Admin - Execute script

In a Nexus Repository Manager cluster, when you create an *Admin - Execute script* task, you are able to specify whether the task runs on all nodes simultaneously or whether the script is only executed on a single node. When a task manipulates a resource that is shared within the cluster, such as repository configuration or a blob store, the task should only be executed on a single node. However, if the task is interacting with a resource that is local to the node, such as the search indices, then the task needs to be configured with *Run task on all nodes in the cluster* selected.

Export Configuration & Metadata for Backup Task

Please see the section on [designing your backup plan](#)(see page 543).

⁸⁷³ <https://support.sonatype.com/hc/en-us/articles/226254487>

⁸⁷⁴ <https://support.sonatype.com/hc/en-us/articles/218501277>

⁸⁷⁵ <https://help.sonatype.com/display/NXRM3M/Load+Balancing#LoadBalancing-MonitoringNodeHealthNEWINRELEASE3.15>

Single-node Tasks

The following Single-node Tasks will run on a single, randomly-selected node in a Nexus Repository Manager cluster:

- *Admin - Cleanup tags*
- *Admin - Compact blob store*
- *Admin - Delete orphaned API keys*
- *Admin - Log database table record counts*
- *Admin - Remove a member from a blob store group*
- *Docker - Delete incomplete uploads*
- *Docker - Delete unused manifests and images*
- *Maven - Delete SNAPSHOT*
- *Maven - Delete unused SNAPSHOT*
- *Maven - Publish Maven Indexer files*
- *Maven - Unpublish Maven Indexer files*
- *Repair - Rebuild Maven repository metadata (maven-metadata.xml)*
- *Repair - Rebuild Yum repository metadata (repodata)*
- *Repair - Reconcile component database from blob store*
- *Repository - Delete unused components*

The following Single-node Tasks will run on a single selected node in a Nexus Repository Manager cluster:

- *Admin - Export database for backup*
- *Repair - Rebuild repository browse*
- *Repair - Reconcile date metadata for blob store*

Multi-node Tasks

The following tasks may run on all nodes or on the current node (i.e. the node the UI is interacting with):

- *Admin - Execute script*
- *Repair - Rebuild repository search*
- *Repair - Reconcile npm /-/v1/search metadata*

JMX Lifecycle Operations

 These operations are intended to be used with the guidance of Sonatype support. Usage without supervision of Sonatype is not supported or recommended.

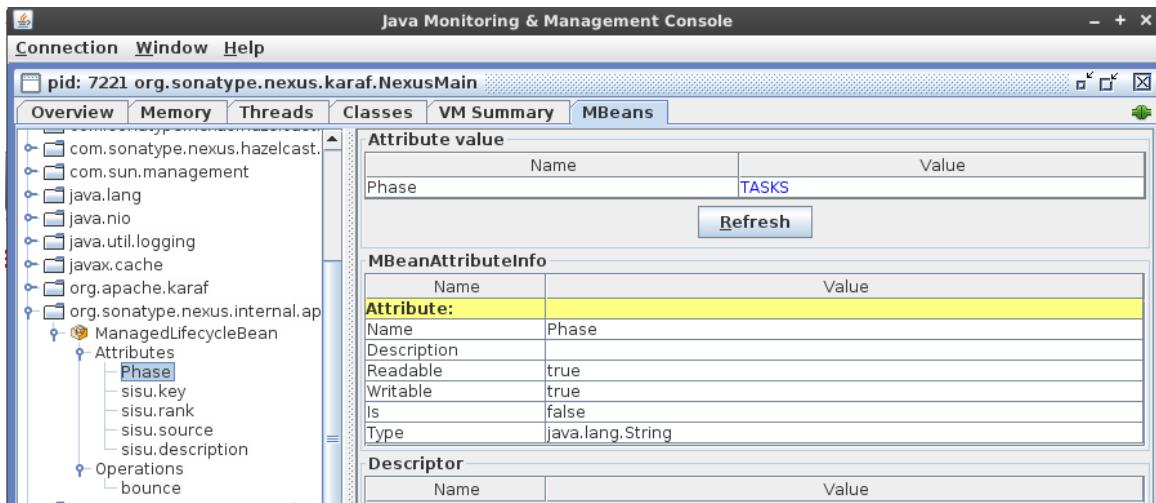
ManagedLifecycleBean

This JMX bean provides a set of attributes and operations to allow the user to inspect and manipulate the current lifecycle phase of a Nexus Repo Manager instance. The expectation is that these will be used for recovery operations. A Phase is a step in the start up process used to group similar components together and ensure that their dependencies are started before them. In order, the phases are: OFF, KERNEL, STORAGE, RESTORE, UPGRADE, SCHEMAS, EVENTS, SECURITY, SERVICES, CAPABILITIES, and TASKS.

1. OFF - NXRM is completely off.
2. KERNEL - The most basic parts of the node are running.
3. STORAGE - The databases and caches are setup.
4. RESTORE - Any restoring from backups happens in this phase.
5. UPGRADE - If any upgrades are needed, they will occur in this phase.
6. SCHEMAS - This creates any missing schemas in the database.
7. EVENTS - The framework that lets services communicate events is started.
8. SECURITY - The parts of NXRM responsible for security are enabled.
9. SERVICES - The bulk of an instances components are started here.
10. CAPABILITIES - The capabilities are started, see [System Configuration](#)(see page 259)
11. TASKS - The task scheduler is enabled, and tasks may run.

Attributes

Phase



The screenshot shows the Java Monitoring & Management Console interface. The title bar reads "Java Monitoring & Management Console". The menu bar includes "Connection", "Window", and "Help". The toolbar has icons for "Overview", "Memory", "Threads", "Classes", "VM Summary", and "MBeans". The main window has a tree view on the left under "pid: 7221 org.sonatype.nexus.karaf.NexusMain" showing various packages like com.sonatype.nexus.hazelcast, com.sun.management, java.lang, java.nio, java.util.logging, javax.cache, org.apache.karaf, and org.sonatype.nexus.internal.ap. Under "org.sonatype.nexus.internal.ap", there is a "ManagedLifecycleBean" node with "Attributes" and "Operations" children. The "Attributes" node has "Phase", "sisu.key", "sisu.rank", "sisu.source", and "sisu.description" listed. The "Operations" node has "bounce" listed. On the right, there are three tables: "Attribute value", "MBeanAttributeInfo", and "Descriptor". The "Attribute value" table shows "Phase" with a value of "TASKS". The "MBeanAttributeInfo" table shows the following attributes for the "Phase" attribute:

Name	Value
Name	Phase
Description	
Readable	true
Writable	true
Is	false
Type	java.lang.String

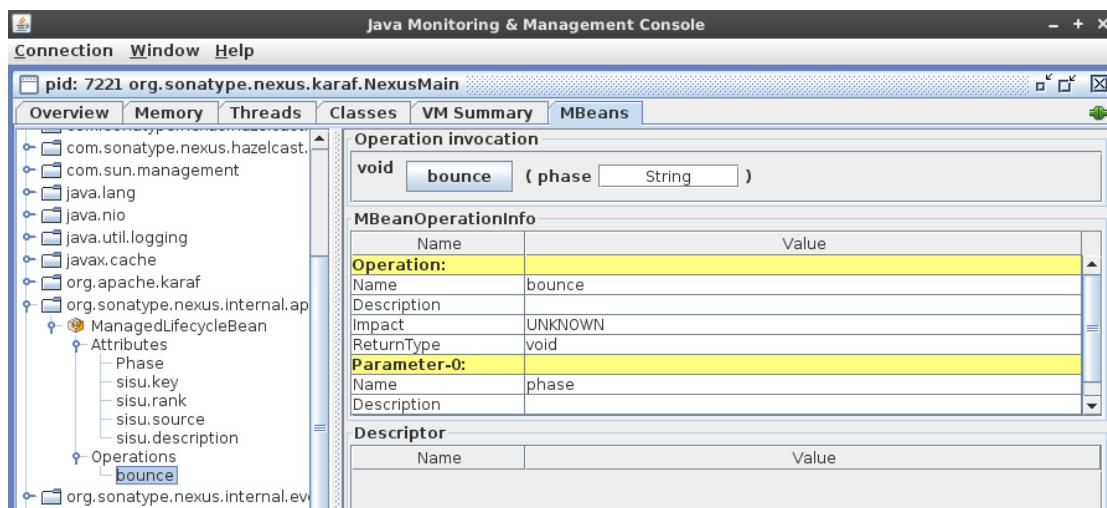
The "Descriptor" table is empty.

Possible Value: Case sensitive string, one of OFF, KERNEL, STORAGE, RESTORE, UPGRADE, SCHEMAS, EVENTS, SECURITY, SERVICES, CAPABILITIES, or TASKS.

This sets NXRM to run in a particular phase. By moving a Nexus Repo manager instance to a new phase, you can start or stop select components. For example, if your instance is running in the TASKS phase and you set the phase to CAPABILITIES, then that instance will not run any scheduled tasks until it is moved back to the TASKS phase.

Operations

Bounce



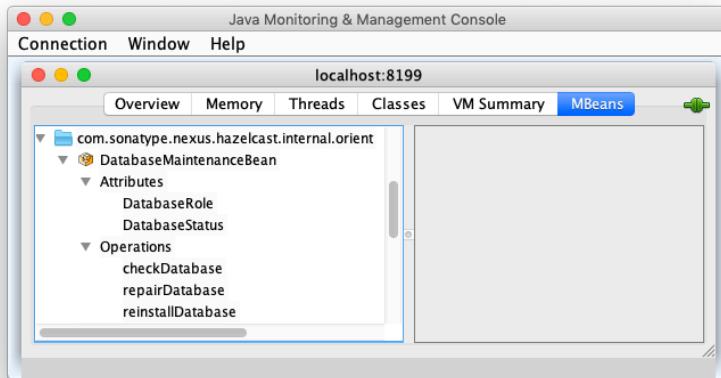
Possible Value for phase: Case sensitive string, one of OFF, KERNEL, STORAGE, RESTORE, UPGRADE, SCHEMAS, EVENTS, SECURITY, SERVICES, CAPABILITIES, or TASKS.

This operation takes the instance down to the supplied Lifecycle phase, and then back to the phase in which it was originally running. This effectively restarts the phases.

JMX Maintenance Operations

 **Available in Nexus Repository Pro**

DatabaseMaintenanceBean



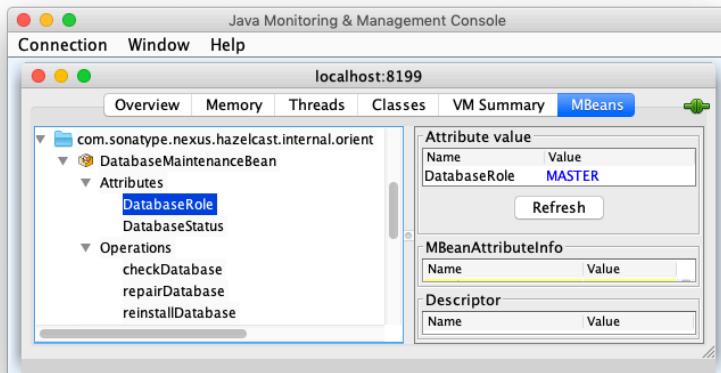
This JMX bean provides a set of attributes and operations to allow the user to inspect and manipulate the state of a node in Orient database cluster. The expectation is that these will be used mainly for troubleshooting and recovery operations.

Attributes

DatabaseRole

 Only available in HA-C

 Applies to all databases on the current node



Possible Values: MASTER, REPLICA

This attribute represents the role of the node in an Orient cluster. The role of **MASTER** represents a fully writable node in an Orient cluster which participates in the writeQuorum. The role of **REPLICA** represents a node in read-only mode which is accepting only idempotent commands, e.g. read and queries.

When setting a node's role to **REPLICA** an override is added to Orient's configuration so that node will always be a **REPLICA**, regardless of the cluster's default role. When the node is set back to **MASTER** that override is removed, so it will defer to the cluster's default role - which is **MASTER** unless it's been frozen.

References

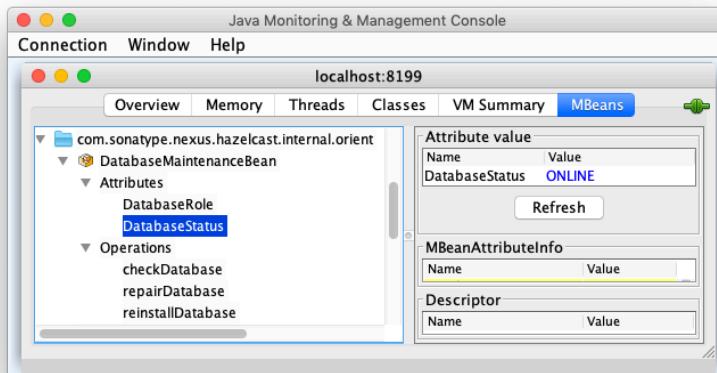
- [Distributed-Architecture: Server Roles⁸⁷⁶](#)

DatabaseStatus

 Only available in HA-C

 Applies to all databases on the current node

⁸⁷⁶ <http://orientdb.com/docs/2.2.x/Distributed-Architecture.html#server-roles>



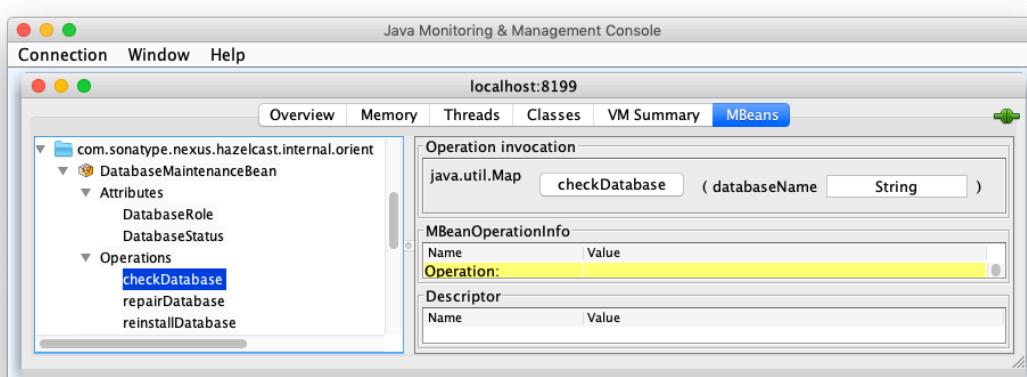
Possible Values: ONLINE, OFFLINE, NOT_AVAILABLE

This attribute represents the status of the node in an Orient cluster. The status of ONLINE represents a node in a normal state which is fully participating in the cluster. The status of OFFLINE represents a node that is not currently participating in the cluster. The status of NOT_AVAILABLE represents a node that is not currently available for some reason.

- ⚠ Setting the database status to NOT_AVAILABLE triggers an automatic delta-sync with the rest of the cluster. If this is successful, it will move back to ONLINE.

Operations

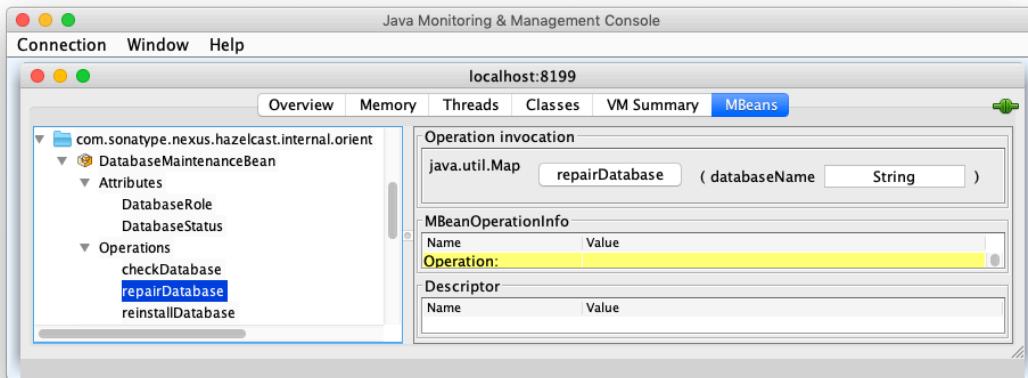
checkDatabase



Possible Database Names: accesslog, component, config, security

Checks database pages for corruption and checks that indices cover all records (i.e. no duplicates).

repairDatabase

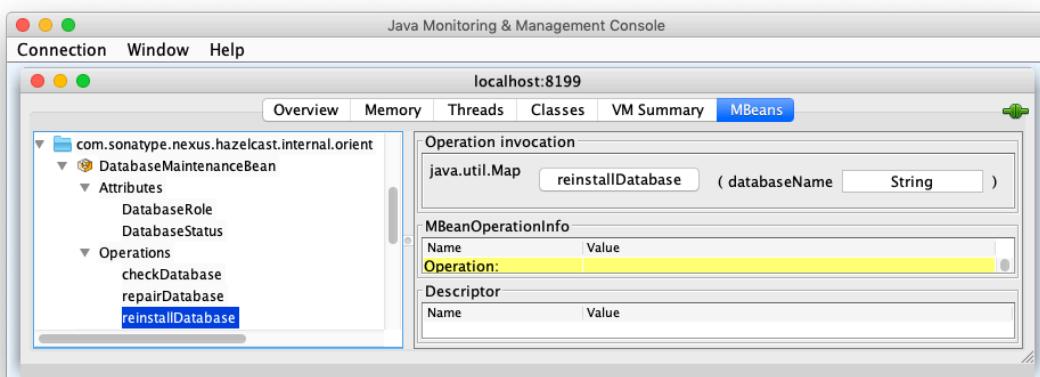


Possible Database Names: accesslog, component, config, security

Attempts to repair any corrupt database pages and rebuilds indices.

reinstallDatabase

 Only available in HA-C



Possible Database Names: accesslog, component, config, security

Attempts to reinstall the full database from the rest of the cluster.

- ⚠** The user can't force a reinstall from a specific node, but typically Orient will choose the node that most recently took a backup or the oldest member (assuming that's not the node requesting the reinstall)

Upgrading your cluster

Upgrading an HA-C Environment from 3.x to 3.y

Prepare for HA-C Upgrade

Do not upgrade unless you have established a reliable and working [HA-C backup and restore](#)(see page 545) process. This includes a backup of shared blob stores filesystems.

HA-C upgrade is not supported and will fail when:

- clustering is enabled (`nexus.clustered=true`)
- databases are read-only
- restoring databases from a backup created with an earlier version

Download, unpack and [pre-configure new version distribution install files on each cluster node](#)⁸⁷⁷ before performing the upgrade.

Perform HA-C Upgrade

We will refer to your three nodes as **NODE-A**, **NODE-B** and **NODE-C**.

NODE-A will be upgraded to a new version in unclustered mode.

NODE-B and **NODE-C** will join and sync databases from **NODE-A** when they start for the first time with the new version.

For each individual node being upgraded, make sure you [adhere to the instructions for upgrading a single node](#)⁸⁷⁸, especially the part of merging configuration file changes

1. Schedule an outage for your service; deny access to all nodes in the cluster using your load balancer.
2. Perform an HA-C backup. [Backup the databases](#)(see page 543) of **NODE-A** and *at least one other node*.

⁸⁷⁷ <https://support.sonatype.com/hc/en-us/articles/115000350007-Upgrading-Nexus-Repository-Manager-3-1-0-and-Newer>

⁸⁷⁸ <https://support.sonatype.com/hc/en-us/articles/115000350007-Upgrading-Nexus-Repository-Manager-3-1-0-and-Newer>

3. Shut down **NODE-C** gracefully and wait for it to stop.
4. Shut down **NODE-B** gracefully and wait for it to stop.
5. Edit **NODE-A** \$data-dir/etc/nexus.properties file and set nexus.clustered=false to disable clustering. Save the file.
6. Shut down **NODE-A** gracefully and wait for it to stop.
7. Start **NODE-A** using the new version. Wait for it to startup in un-clustered mode and perform automatic upgrade steps.
8. Verify **NODE-A** is functioning after the upgrade in un-clustered mode, e.g. use the UI and access repository content.
9. If **NODE-A** fails verification, follow our [HA-C upgrade rollback procedure](#)(see page 564).
10. If **NODE-A** passes verification, then delete the **NODE-B** and **NODE-C** \$data-dir/db directory (you have already made database backups earlier).
11. Edit **NODE-A** \$data-dir/etc/nexus.properties file and set nexus.clustered=true to enable clustering on the next startup. Save the file.
12. Restart **NODE-A**. Wait for it to startup in clustered mode.
13. Start **NODE-B** with the new version. Wait for it to startup in HA-C mode and sync the databases from **NODE-A**.
14. Start **NODE-C** with the new version. Wait for it to startup in HA-C mode and sync the databases from **NODE-A** or **NODE-B**.
15. Sign-in as an administrator and confirm that all nodes are visible in the *System → Nodes* administration page and that there are no status warnings about cluster health.
16. Verify that all nodes are functioning normally on the new version, e.g. use the UI and access repository content.
17. Restore access through your load balancer and resume operations.

Rollback a Failed HA-C Upgrade

Do not attempt to start a older version of NXRM against an already upgraded or partially upgraded [data directory](#)(see page 183) or databases. This will fail.

The only supported way to rollback to an old version is to start NXRM against a set of databases and data directory were created using an identical NXRM version.

Load Balancing

Overview

Load balancing is required to achieve transparent redundancy in a High Availability Cluster. The following outlines considerations relating specifically to NXRM.

Ports

NXRM listens on port 8081 for HTTP by default. There is no default port for HTTPS. These ports are also configurable (see [NXRM Port Configuration](#)(see page 188)).

Docker Repositories

The docker client does not allow a context as part of the path to a registry, as the namespace and image name are embedded in the URLs it uses. NXRM allows custom ports to be exposed to create a listener on a root context. See [SSL and Repository Connector Configuration](#)(see page 384) for more details. Alternatively you may choose to use a [URL rewriting scheme](#)⁸⁷⁹ to achieve the same results without needing to configure custom ports (not covered here). Using URL rewriting allows you to sidestep this limitation, otherwise exposing and using the configured port(s) are required.

Sticky Sessions

The NXRM UI uses a cookie named NXSESSIONID to maintain a users state. The load balancer should be configured to enforce sticky sessions for requests containing this cookie.

Monitoring Node Health

NEW IN RELEASE 3.15.0

NXRM provides two endpoints you can use to check the status of your NXRM node. The *Readable Health Check* endpoint verifies that a node can handle read requests. The *Writable Health Check* endpoint verifies that a node can handle read and write requests. Since a node that can handle read requests but not write requests may still be considered serviceable, it is left to the user to decide which status endpoint to use.

These endpoints can be used by a load balancer to determine if a node should be considered to handle requests. For both status endpoints, Success is represented as HTTP 200 OK. Failure is represented as HTTP 503 SERVICE UNAVAILABLE. The nexus.log file for the node should be inspected for further details.

The URLs for the Health Check endpoints are:

Name	URL
Readable Health Check	<code>http://<hostname>:<port>/service/rest/v1/status</code>
Writable Health Check <small>NEW IN RELEASE 3.16.0</small>	<code>http://<hostname>:<port>/service/rest/v1/status/writable</code>

⁸⁷⁹ <https://support.sonatype.com/hc/en-us/articles/360000761828>

SSL Termination

NXRM can be configured to serve content using SSL (see [Configuring to Serve Content via HTTPS\(see page 344\)](#)). The recommended approach for HA-C is to use the load balancer for SSL termination as it alleviates the need to configure certificates on each NXRM node.

sitemap.xml

Quick Start Guide - Proxying Maven and NPM

If you're new to repository management with Nexus Repository Manager 3, use this guide to get familiar with configuring the application as a dedicated proxy server for Maven and npm builds. To reach that goal, follow each section to:

- install Nexus Repository Manager 3
- run the repository manager locally
- proxy a basic Maven and npm build

When you complete the steps, the components will be cached locally to your repository manager. After meeting the [requirements\(see page 566\)](#) to run the repository manager, it should take approximately 15 minutes to proxy Maven and npm with the code snippets below.

Requirements

Before you can set up the proxy server for Maven and npm, you'll need to install and configure the following external tools for the repository manager:

- Java 8 Development Kit (JDK) - the repository manager is a Java server application. As explained in [System Requirements\(see page 141\)](#), we strongly recommend using [Java 8⁸⁸⁰](#) to ensure effective runtime of Nexus Repository Manager 3.
- Apache [Maven⁸⁸¹](#) - when downloaded, Nexus Repository Manager 3 includes access to open source components from the [Central Repository⁸⁸²](#) by default. These components are defined by both a [settings.xml⁸⁸³](#) file and [POMs⁸⁸⁴](#) which maintain information on projects and dependencies.

⁸⁸⁰ <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

⁸⁸¹ <https://maven.apache.org/download.cgi>

⁸⁸² <https://search.maven.org/>

⁸⁸³ <https://maven.apache.org/settings.html>

⁸⁸⁴ <https://maven.apache.org/pom.html>

- [npm](#)⁸⁸⁵ - a popular format supported by the repository manager. Unlike Maven, Nexus Repository Manager 3 doesn't currently ship with a default npm repository configuration. These components are configured with an `.npmrc` file.

Part 1 - Installing and Starting Nexus Repository Manager 3

1. Create an [installation directory](#)⁸⁸⁶ in your desired location.
2. [Download the most recent repository manager](#)(see page 22) for your operating system.
3. If the file is downloaded to a location outside the installation directory, move it there.
4. Unpack the `.tar.gz` or `.zip` file in its new location. Both an application (i.e. `nexus-<version>`) and data directory (i.e. `.../sonatype-work/nexus3`) are created after extraction.
5. Go to the application directory which contains the repository manager file you need to start up.
6. In the application directory, run the startup script launching the repository manager:
 - Linux or Mac: `./bin/nexus run`
 - Windows: `bin/nexus.exe /run`

Wait until the log shows the message *Started Sonatype Nexus*.

7. Open your browser and type `http://localhost:8081/` in your URL field.
8. From the user interface click *Sign In*, which generates a modal to enter your credentials.
9. Log in with the default credentials `admin / admin123`, the respective username / password.
10. [Change the default password](#)(see page 229)!

Part 2 - Proxying Maven and npm Components

When you proxy components the repository manager acts as a local intermediary server for any download requests going to remote repositories / registries. After logging in, these next steps will show you how to configure then test your configuration with local builds for a Maven and npm project.

Maven

 If you have an existing Maven configuration file (`settings.xml`) that you want to retain, back it up before doing any modifications.

1. In your file system, open your `settings.xml` and change the contents of the snippet below. You can find this file in `.m2`, e.g `~/ .m2/settings.xml`.

885 <https://www.npmjs.com/get-npm>

886 <https://support.sonatype.com/hc/en-us/articles/231742807#InstallDir>

```
<settings>
  <mirrors>
    <mirror>
      <!--This sends everything else to /public -->
      <id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://localhost:8081/repository/maven-proxy/</url>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>nexus</id>
      <!--Enable snapshots for the built in central repo to direct -->
      <!--all requests to nexus via the mirror -->
      <repositories>
        <repository>
          <id>central</id>
          <url>http://central</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>http://central</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
  <activeProfiles>
    <!--make the profile active all the time -->
    <activeProfile>nexus</activeProfile>
  </activeProfiles>
</settings>
```

2. Go to the repository manager user interface.



3. Click the *Administration* button in the left navigational menu, then click .

4. Click the *Create repository* button and choose the *maven2 (proxy)* recipe from the list.

5. Add the following text in the required fields:

- *Name*: maven-proxy
- *Remote storage URL*: <https://repo1.maven.org/maven2>

6. Click *Create repository* to complete the form.

7. From the command-line interface, create the POM file (*pom.xml*) with the values below:

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>nexus-proxy</artifactId>
  <version>1.0-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.10</version>
    </dependency>
  </dependencies>
</project>
```

- Run the Maven build with the command `mvn package`.

⚠ If you want to view the details of your Maven test build, skip to [Part 3\(see page 570\)](#). However, if you want to test an [npm\(see page 569\)](#) build, continue to the next section in this part.

npm

⚠ If you have an existing npm configuration file (`.npmrc`) that you want to retain, back it up before doing any modifications.

- Click the *Administration* button  in the left navigational menu, then click .
- Click the *Create repository* button and choose *npm (proxy)* from the list.
- Add the following text in the required fields:
 - Name* : npm-proxy
 - Remote Store URL* : <https://registry.npmjs.org/>
- From the command-line interface run `npm config set registry http://localhost:8081/repository/npm-proxy`
- From the command-line interface, create a package.json with the values below:

```
{  
  "name": "sample_project1",  
  "version": "0.0.1",  
  "description": "Test Project 1",  
  "dependencies" : {  
    "commonjs" : "0.0.1"  
  }  
}
```

6. Run the npm build with the command `npm install`.

Part 3 - Viewing Proxied Components

After your Maven and npm projects are successfully built, follow these steps to view the cached components:



1. Click the *Browse* button , from the main toolbar.
2. Click *Components*.
3. Of your components, choose *maven-proxy* or *npm-proxy*. You'll see the test component you proxied for the respective format during the previous build steps.
4. Click on a component name to review its details.

The *Components* screen is a sub-section to the *Browse* interface. So in order to view other components, click *Components* directly from the current screen and select another repository name from step 3 in this part.