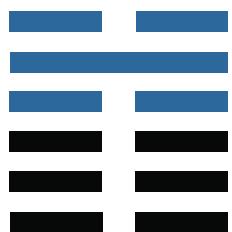


Edition 3.2

Nexus 2



Nexus

Repository Management with Nexus

Repository Management with Nexus

Manfred Moser

Tim O'Brien

Damian Bradicich

John Casey

Tamás Cservenák

Brian Demers

Brian Fox

Marvin Froeder

Anders Hammar

Rich Seddon

Juven Xu

Jason van Zyl

Repository Management with Nexus

Ed. 3.0.2

Contents

1	Introducing Sonatype Nexus	1
1.1	Introduction	1
1.2	Nexus Open Source	1
1.2.1	Nexus Open Source Features	2
1.2.2	Nexus Open Source License	3
1.3	Nexus Professional	3
1.3.1	Nexus Professional Features	3
1.3.2	Nexus Professional License	5
1.4	Choosing a Nexus Edition	6
1.4.1	Use Nexus Open Source	6
1.4.2	Use Nexus Professional	7
1.4.3	Comparing Nexus Open Source and Nexus Professional Features	8

1.5	History of Nexus	9
2	Repository Management	10
2.1	Repository Management	10
2.1.1	Proxying Public Repositories	10
2.1.2	Managing Releases and Snapshots	11
2.1.3	Getting Control of Dependencies	11
2.1.4	A Nexus for Collaboration	12
2.2	What is a Repository?	12
2.2.1	Release and Snapshot Repositories	13
2.2.2	Repository Coordinates	14
2.2.3	Addressing Resources in a Repository	15
2.2.4	The Maven Central Repository	15
2.3	What is a Repository Manager	16
2.3.1	Core Capabilities of a Repository Manager	17
2.3.2	Additional Features of a Repository Manager	18
2.4	Reasons to Use a Repository Manager	19
2.4.1	Speed Up Your Builds	19
2.4.2	Save Bandwidth	19

2.4.3	Ease the Burden on Central	20
2.4.4	Gain Predictability and Scalability	20
2.4.5	Control and Audit Dependencies and Releases	20
2.4.6	Deploy 3rd Party Artifacts	21
2.4.7	Collaborate with Internal Repositories	21
2.4.8	Distribute with Public Repositories	21
2.5	Adopting a Repository Manager	22
2.5.1	Stage Zero: Before Using a Repository Manager	22
2.5.2	Stage One: Proxying Remote Repositories	23
2.5.3	Stage Two: Hosting a Repository Manager	23
2.5.4	Stage Three: Continuous Collaboration	25
2.5.5	Stage Four: Life-cycle Integration	25
3	Installing and Running Nexus	27
3.1	Nexus Prerequisites	27
3.2	Downloading Nexus	27
3.2.1	Downloading Nexus Open Source	28
3.2.2	Downloading Nexus Professional	29
3.3	Installing Nexus	29

3.4	Upgrading Nexus	31
3.5	Running Nexus	32
3.6	Post-Install Checklist	34
3.6.1	Step 1: Change the Administrative Password and Email Address	34
3.6.2	Step 2: Configure the SMTP Settings	34
3.6.3	Step 3: Enable Remote Index Downloads	35
3.6.4	Step 4: Change the Deployment Password	35
3.6.5	Step 5: Install Professional License (Nexus Professional Only)	36
3.6.6	Step 6: If necessary, set the LANG Environment Variable	36
3.7	Configuring Nexus as a Service	36
3.7.1	Running as a Service on Linux	36
3.7.1.1	Add Nexus as a Service on Red Hat, Fedora, and CentOS	37
3.7.1.2	Add Nexus as a Service on Ubuntu and Debian	38
3.7.2	Running as a Service on Mac OS X	38
3.7.3	Running as a Service on Windows	39
3.8	Running Nexus Behind a Proxy	39
3.9	Installing the Nexus WAR	40
3.9.1	Running the Nexus WAR in Glassfish	41

3.10	Installing a Nexus Professional License	41
3.10.1	Evaluation Expiration	45
3.11	Nexus Directories	45
3.11.1	Sonatype Work Directory	45
3.11.2	Nexus Configuration Directory	49
3.12	Installing Additional Plugins	50
4	Configuring Maven to Use Nexus	52
4.1	Introduction	52
4.2	Configuring Maven to Use a Single Nexus Group	52
4.3	Adding Custom Repositories for Missing Dependencies	54
4.4	Adding a New Repository	54
4.5	Adding a Repository to a Group	55
5	Using Nexus	58
5.1	Introduction	58
5.2	Browsing Repositories	59
5.2.1	Viewing the Artifact Information	61
5.2.2	Viewing the Artifact Maven Information	62
5.2.3	Using the Artifact Archive Browser	63

5.2.4	Viewing the Artifact Dependencies	63
5.3	Browsing Groups	64
5.4	Searching for Artifacts	66
5.4.1	Search Overview	66
5.4.2	Advanced Search	68
5.4.3	Nexus OpenSearch Integration	70
5.5	Uploading Artifacts	72
5.6	Browsing System Feeds	73
5.7	System Files	75
5.8	Changing Your Password	76
5.9	Filing a Problem Report	77
6	Configuring Nexus	78
6.1	Customizing Server Configuration	78
6.1.1	SMTP Settings	79
6.1.2	HTTP Request Settings	79
6.1.3	Security Settings	80
6.1.4	Error Reporting Settings	81
6.1.5	Application Server Settings	83

6.1.6	Default HTTP Proxy Settings	84
6.1.7	System Notification Settings	85
6.1.8	New Version Availability	85
6.1.9	PGP Key Server Information	85
6.2	Managing Repositories	86
6.2.1	Proxy Repository	86
6.2.2	Hosted Repository	87
6.2.3	Virtual Repository	87
6.2.4	Configuring Repositories	87
6.2.5	Selecting Mirrors for Proxy Repositories	92
6.2.6	Adding a Mirror Entry for a Hosted Repository	92
6.2.7	Viewing Repository Summary Panel	93
6.2.8	Auto Block/Unblock of Remote Repositories	95
6.3	Managing Groups	95
6.4	Managing Routes	97
6.5	Managing Scheduled Tasks	99
6.5.1	Managing Configuration Backups with a Scheduled Task	102
6.6	Managing Security	104

6.7	Managing Privileges	105
6.8	Managing Repository Targets	107
6.9	Managing Roles	109
6.10	Managing Users	114
6.11	Network Configuration	116
6.12	Nexus Logging Configuration	117
6.13	Nexus Plugins and REST Interfaces	118
7	Nexus Smart Proxy	120
7.1	Introduction	120
7.2	Enabling Smart Proxy Publishing	121
7.3	Establishing Trust	122
7.4	Repository Specific Smart Proxy Configuration	123
7.5	Smart Proxy Security and Messages	125
7.6	Example Setup	126
8	Nexus LDAP Integration	127
8.1	Introduction	127
8.2	Enabling the LDAP Authentication Realm	127
8.3	Configuring Nexus LDAP Integration	129

8.4	Connection and Authentication	130
8.5	User and Group Mapping	132
8.6	Mapping Users and Groups with Active Directory	135
8.7	Mapping Users and Groups with posixAccount	137
8.8	Mapping Roles to LDAP Users	138
8.9	Mapping Nexus Roles for External Users	139
8.10	Mapping External Roles to Nexus Roles	143
8.11	Enterprise LDAP Support	146
8.11.1	Support for Multiple Servers and LDAP Schemas	148
8.11.2	Enterprise LDAP Performance Caching and Timeout	149
8.11.3	User and Group Templates	150
8.11.4	Testing a User Login	151
9	Nexus Procurement Suite	152
9.1	Introduction	152
9.2	The Stages of Procurement	153
9.3	Two Approaches to Procurement	153
9.3.1	Procured Release Repository	153
9.3.2	Procured Development Repository	154

9.4 Accessing Artifact Procurement	155
9.5 Setting up a Procured Repository	155
9.5.1 Enable Remote Index Downloads	156
9.5.2 Create a Hosted Repository	157
9.5.3 Configuring Procurement for Hosted Repository	158
9.5.4 Procured Repository Administration	160
9.6 Configuring a Procurement Rule	161
9.7 Managing Procurement Rules	164
9.8 Stopping Procurement	165
10 Build Promotion with the Nexus Staging Suite	167
10.1 Introduction	167
10.1.1 Releasing Software without a Staging Repository	167
10.1.2 How the Staging Suite Works	168
10.1.3 Multi-level Staging and Build Promotion	170
10.2 Using the Nexus Staging Suite	171
10.2.1 Configuring Staging Profiles	171
10.2.2 Configuring Build Promotion Profiles	177
10.2.3 Adding the Staging Deployer Role	179

10.3 Performing a Staged Deployment with Maven	181
10.3.1 Creating a New Project	181
10.3.2 Update the POM Deployment Configuration	182
10.3.3 Update settings.xml with Deployment Credentials	183
10.3.4 Deploying to a Staged Repository	184
10.4 Manually Uploading a Staged Deployment in Nexus	185
10.5 Managing Rulesets	187
10.5.1 Managing Staging Rulesets	187
10.5.2 Defining Rulesets for Promotion	189
10.6 Managing Staging Repositories in Nexus	190
10.6.1 Closing an Open Repository	191
10.6.2 Using the Staging Repository	193
10.6.3 Releasing a Staging Repository	197
10.6.4 Promoting a Staging Repository	199
10.6.5 Releasing, Promoting, and Dropping Build Promotion Profiles	201
10.7 Managing Staging Repositories with the Nexus Maven Plugin	203
10.7.1 Running the Nexus Maven Plugin	203
10.7.2 Configuring Nexus Maven Plugin for Staging	204

10.7.3 Listing Your Open Staging Repositories	205
10.7.4 Closing a Staging Repository	206
10.7.5 Dropping a Closed Staging Repository	207
10.7.6 Promoting a Closed Staging Repository	208
11 Repository Health Check	210
12 Managing Maven Settings	215
12.1 Introduction	215
12.2 Manage Maven Settings Templates	215
12.3 Downloading Maven Settings with the Nexus Maven Plugin	217
12.3.1 Running the Nexus Maven Plugin	217
12.3.2 Configuring Nexus Maven Plugin for Settings Management	218
12.3.3 Downloading Maven Settings	219
13 OSGi Bundle Repositories	221
13.1 Introduction	221
13.2 Proxy OSGi Bundle Repositories	222
13.3 Hosted OSGi Bundle Repositories	224
13.4 Virtual OSGi Bundle Repositories	225

13.5 Grouping OSGi Bundle Repositories	226
14 P2 Repositories	228
14.1 Introduction	228
14.2 Proxy P2 Repositories	228
14.3 Grouping P2 Repositories	229
15 .NET Package Repositories	231
15.1 Introduction	231
15.2 NuGet Proxy Repositories	232
15.3 NuGet Hosted Repositories	234
15.4 NuGet Virtual Repositories	235
15.5 NuGet Group Repositories	236
15.6 Accessing Packages in Repositories and Groups	237
15.7 Deploying Packages to NuGet Hosted Repositories	238
15.7.1 Creating a NuGet API-Key	238
15.7.2 Creating a Package for Deployment	238
15.7.3 Deployment with the NuPkg Upload User Interface	239
15.7.4 Command line based Deployment to a Nexus NuGet Hosted Repository	239
15.8 Integration of Nexus NuGet Repositories in Visual Studio	239

16 Deploying Sites to Nexus	241
16.1 Introduction	241
16.2 Creating a New Maven Project	241
16.3 Configuring Maven for Site Deployment	243
16.4 Adding Credentials to Your Maven Settings	244
16.5 Creating a Maven Site Repository	245
16.6 Add the Site Deployment Role	246
16.7 Publishing a Maven Site to Nexus	248
17 Custom Metadata Plugin	250
17.1 Introduction	250
17.2 Viewing Artifact Metadata	250
17.3 Editing Artifact Metadata	251
17.4 Searching Artifact Metadata	252
18 User Account Plugin	255
18.1 Introduction	255
18.2 Installing the User Account Plugin	255
18.3 Configuring the User Account Plugin	256
18.4 Signing Up for an Account	257

18.5 Manual Activation of New Users	259
18.6 Modifying Default User Permissions	260
19 Nexus Atlassian Crowd Plugin	263
19.1 Introduction	263
19.2 Installing the Crowd Plugin	263
19.3 Configuring the Crowd Plugin	264
19.4 Crowd Access Settings	265
19.4.1 Crowd HTTP Settings	266
19.4.2 Crowd HTTP Proxy Settings	267
19.4.3 Miscellaneous Settings	267
19.5 Adding the Crowd Authentication Realm	268
19.6 Configuring a Nexus Application in Crowd	268
19.7 Mapping Crowd Groups to Nexus Roles	271
19.8 Adding a Crowd Role to a Nexus User	273
20 Artifact Bundles	276
20.1 Introduction	276
20.2 Creating an Artifact Bundle from a Maven Project	277
20.3 Uploading an Artifact Bundle to Nexus	280

21 Nexus Best Practises	284
21.1 Introduction	284
21.2 Repositories per Project/Team	284
21.3 Partition Shared Repositories	285
21.3.1 Selecting an Approach	285
22 Developing Nexus Plugins	287
22.1 Nexus Plugins	288
22.1.1 Nexus Plugin API	288
22.2 Nexus Extension Points	289
22.3 Nexus Plugin Extension Points	289
22.4 Nexus Plugin Extension	290
22.5 Nexus Index HTML Customizer	290
22.6 Static Plugin Resources	290
22.7 Plugin Templates	291
22.8 Event Inspectors	291
22.9 Content Generators	291
22.10 Content Classes	292
22.11 Storage Implementations	292

22.12 Repository Customization	292
22.13 Item and File Inspectors	293
22.14 Nexus Feeds	293
22.15 Nexus Tasks and Task Configuration	293
22.16 Application Customization	294
22.17 Request Processing	294
22.18 Using the Nexus Plugin Archetype	294
22.19 Set the Target Nexus Version	297
22.20 Building a Nexus Plugin Project	298
22.21 Creating a Complex Plugin	298
22.22 Nexus Plugin Descriptor Maven Plugin	302
22.23 The Nexus Plugin Descriptor	303
22.24 Defining Custom Repository Types	304
A Migrating to Nexus from Artifactory	305
A.1 Introduction	305
A.2 Creating an Artifactory Backup	305
A.3 Installation of the Migration Plugin and Artifactory Bridge	307
A.4 Importing an Artifactory System Backup	309

A.5 Configuring the Artifactory Import	310
A.6 Configuring Artifactory Group Imports	311
A.7 Configuring Artifactory Repository Imports	312
A.8 Configuring Users and Privileges in the Artifactory Import	314
A.9 Performing the Artifactory Import	315
A.10 Configuring Artifactory Clients to Use Nexus	316
B Migrating to Nexus from Archiva	318
B.1 Introduction	318
B.2 Migrating Archiva Repositories	318
B.3 Migrating an Archiva Managed Repository	318
B.4 Migrating an Archiva Proxy Connector	322
C Configuring Nexus for SSL	329
C.1 Introduction	329
C.2 Importing a SSL Client Certificate	329
C.2.1 Downloading the SSL Import Tool	330
C.2.2 Importing a Client Certificate	330
C.2.3 Import the Server SSL Chain	330
C.2.4 Import the Client SSL Key/Certificate Pair	331

C.2.5 Configuring Nexus Start-up	331
C.3 Configuring Nexus to Serve SSL	332
C.3.1 Configure the Java Keystore	333
C.3.2 Configure Nexus/Jetty to Use the New Keystore	333
C.3.3 Modify the application-port for SSL connections	334
C.4 Redirecting Non-SSL Connections to SSL	335
D Contributing to the Nexus Book	338
D.1 Contributor License Agreement (CLA)	338
D.2 Contributors, Authors, and Editors	339
D.3 Tools Used to Build and Write this Book	339
D.4 How to Build the Book	341
D.5 Subscribing to the Book Developers List	341
E Copyright	343
F Creative Commons License	345
F.1 Creative Commons BY-NC-ND 3.0 US License	346
F.2 Creative Commons Notice	350

Chapter 1

Introducing Sonatype Nexus

1.1 Introduction

Nexus manages software "artifacts" required for development, deployment, and provisioning. If you develop software, Nexus can help you share those artifacts with other developers and end-users. Maven's central repository has always served as a great convenience for users of Maven, but it has always been recommended to maintain your own repositories to ensure stability within your organization. Nexus greatly simplifies the maintenance of your own internal repositories and access to external repositories. With Nexus you can completely control access to, and deployment of, every artifact in your organization from a single location.

1.2 Nexus Open Source

Nexus Open Source provides you with an essential level of control over the external Maven repositories you use and the internal repositories you create. It provides infrastructure and services for organizations that use repository managers to obtain and deliver software. If you create software libraries or applications for your end-users, you can use Nexus Open Source to distribute your software. If your software depends upon open source software components, you can cache software artifacts from remote repositories.

1.2.1 Nexus Open Source Features

Hosting Repositories

When you host a Maven repository with Nexus Open Source, you can upload artifacts using the Nexus interface, or you can deploy artifacts to hosted repositories using Maven. Nexus will also create the standard Nexus Index for all of your hosted repositories which will allow tools like m2eclipse to rapidly locate software artifacts for your developers.

Proxy Remote Repositories

When you proxy a remote repository with Nexus Open source you can control all aspects of the connection to a remote repository including security parameters, HTTP proxy settings. You can configure which mirrors Nexus will download artifacts from, and you can control how long Nexus will store artifacts and how it will expire artifacts which are no longer referenced by your build.

Repository Groups

Grouping repositories allows you to consolidate multiple repositories into a single URL. This makes configuring your development environment very easy. All of your developers can point to a single repository group URL, and if anyone ever needs a custom remote repository added to the group, you can do this in a central location without having to modify every developer's workstation.

Fine-grained Security Model

Nexus Open Source ships with a very capable and customizable security model. Every operation in Nexus is associated with a privilege, and privileges can be combined into standard Nexus roles. Users can then be assigned both individual privileges and roles that can be applied globally or at a fine grained level. You can create custom administrative roles that limit certain repository actions such as deployment to specific groups of developers and you can use these security roles to model the structure of your organization.

Flexible LDAP Integration

If your organization uses an LDAP server, Nexus Professional can integrate with an external authentication and access control system. Nexus Professional is smart enough to be able to automatically map LDAP groups to the appropriate Nexus roles, and it also provides a very flexible facility for mapping existing users and existing roles to Nexus roles.

Artifact Search

Nexus Open Source provides an intuitive search feature which allows you to search for software artifacts by identifiers such as groupId, artifactId, version, classifier, and packaging, names of classes contained in Java archives, keywords, and artifact checksums. Nexus search makes use of the industry standard for repository indexes, the Nexus Index format, and Nexus will automatically download a Nexus index from all remote repositories which create a Nexus index. Nexus will also automatically expose a Nexus index for any hosted repositories you create.

Scheduled Tasks

Nexus Open Source has the concept of scheduled tasks: periodic jobs which take care of various repository management tasks such as deleting old snapshots, evicting unused items, and publishing repository indexes.

REST Services

Nexus Open Source is based on a series of REST services, and when you are using the Nexus web front-end UI, you are really just interacting with a set of REST service. Because of this open architecture, you can leverage the REST service to create custom interactions or to automate repository management with your own scripts.

Nexus Plugins

Nexus Open Source provides a rich API for extension in the form of Nexus Plugins. When you write a Nexus Plugin, you can customize REST services, the Nexus UI, repository formats, or write components that can intercept requests and add new capabilities to the platform. The plugin API which you have access to in Nexus Open Source is the same plugin API that is used to implement value-added features available in Nexus Professional.

Integration with m2eclipse

When you use Nexus as a repository manager it creates indexes that support some of the next-generation tools available in m2eclipse - Sonatype's Maven plugin for the Eclipse IDE. If you publish new artifacts and archetypes to Nexus, they are immediately available to m2eclipse project creation wizards and are included in m2eclipse search results.

1.2.2 Nexus Open Source License

Nexus Open Source is made available under the Eclipse Public License version 1.0. The text of this license is available from the Open Source Initiative (OSI) here: <http://www.opensource.org/licenses/eclipse-1.0.php>

1.3 Nexus Professional

Nexus Professional was designed to meet the needs of the enterprise. It is a central point of access to external repositories which provides the necessary controls to make sure that only approved artifacts enter into your software development environment. It is also a central distribution point with the intelligence required to support the decision that go into making quality software. The extensibility provided by the custom metadata plugin coupled with REST services only available in Nexus Professional also lay the foundation for highly complex interactions within the enterprise. Once you start to use the work-flow and decision support features of Nexus Professional, you will start to see it as the "assembly line" - the central collaboration point for your software development efforts.

1.3.1 Nexus Professional Features

Nexus Procurement Suite

Consider the default behaviour of a proxy repository. Any developer can reference any artifact stored in a remote repository and cause Nexus to retrieve the artifact from the remote repository and serve back to a developer. Very often a company might want to control the set of artifacts which can be referenced in a proxy repository. Maybe the company has unique security requirements which require every third-party library to be subjected to a rigorous security audit before they can be used. Or, maybe another company has a legal team which needs to verify that every artifact referenced by your software adheres to an inflexible set of license guidelines. The Nexus Procurement Suite was designed to give organization this level of control over the artifacts that can be served from Nexus.

Nexus Staging Suite

When was the last time you did a software release to a production system? Did it involve a QA team that had to sign-off on a particular build? What was the process you used to re-deploy a new build if QA found a problem with the system at the last minute? Because few organizations use a mature process to manage binary software artifacts, there is little in the way of infrastructure designed to keep track of the output of a build. The Nexus Staging Suite changes this by providing work-flow support for binary software artifacts. If you need to create a release artifact and deploy it to a hosted repository, you can use the Staging Suite to post a collection of related, staged artifacts which can be tested, promoted, or discarded as a unit. Nexus keeps track of the individuals that are involved in a staged, managed release and can be used to support the decisions that go into producing quality software.

Hosting Project Web Sites

Nexus Professional is a publishing destination for project web sites. While you very easily generate a project web site with Maven, without Nexus, you will need to set up a WebDAV server and configure both your web server and build with the appropriate security credentials. With Nexus, you can deploy your project's web site to the same infrastructure that hosts the project's build output. This single destination for binaries and documentation helps to minimize the number of moving parts in your development environment. You don't have to worry about configuring another web server or configuring your builds to distribute the project site using a different protocol, you simply point your project at Nexus and deploy the project site.

Support for OSGi Repositories

Instead of just supporting Maven repositories, Nexus Professional supports OSGi Bundle repositories and P2 repositories for those developers who are targeting OSGi or the Eclipse platform. Just like you can proxy, host, and group Maven repositories, Nexus Professional allows you to do the same with OSGi repositories.

Enterprise LDAP Support

Nexus Professional offers LDAP support features for enterprise LDAP deployments including detailed configuration of cache parameters, support for multiple LDAP servers and backup mirrors, the ability to test user logins, support for common user/group mapping templates, and the ability to support more than one schema across multiple servers.

Support for Atlassian Crowd

If your organization uses Atlassian Crowd, Nexus Professional can delegate authentication and access control to a Crowd server and map Crowd groups to the appropriate Nexus roles.

The User Account Plugin

When you are running a large, public instance of Nexus, it is often very useful to allow users to sign up for an account without the assistance of an administrator. Nexus Professional's User Account plugin allows for just this. With this plugin activate, a new user simply has to fill out a simple form and type in letters from a captcha. Once a user has signed up for Nexus, Nexus will then send an email with a validation link. If you are working in an environment with hundreds or thousand of users the user account plugin will allow you to support the tool without having to create logins for each individual user.

Maven Settings Management

Nexus Professional along with the Nexus Maven Plugin allow you to manage Maven Settings. Once you have developed a Maven Settings template, developers can then connect to Nexus Professional using the Nexus Maven plugin which will take responsibility for downloading a Maven Settings file from Nexus and replacing the existing Maven Settings on a local workstation.

Support for Artifact Bundles

When software is deployed to the Maven Central repository, it is deployed as a signed artifact bundle. Nexus Professional's Staging Suite allows you to upload artifact bundles to a staged repository.

Artifact Validation and Verification

The software artifacts you download from a remote repository are often signed with PGP signatures. Nexus Professional will make sure that these PGP signature are valid and the procurement plugin defines a few other rules that can be applied to artifacts which are downloaded from remote repositories. Nexus Professional also defines an API which allows you to create your own custom verification rules.

Custom Repository Metadata

Nexus Professional provides a facility for user-defined, custom metadata. If you need to keep track of custom attributes to support approval work-flow or to associate custom identifiers with software artifacts, you can use Nexus to define and manipulate custom attributes which can be associated with artifacts in a Nexus repository.

1.3.2 Nexus Professional License

Nexus Professional is made available under a commercial license for businesses. Is available for free for use in qualifying Open Source projects, and is available at a discount for select Non-profits.

1.4 Choosing a Nexus Edition

If you are wondering which edition is appropriate for your organization. The following sections outline some reasons for choosing either Nexus Open Source or Nexus Professional.

1.4.1 Use Nexus Open Source...

...if you are new to Repository Management

If you are new to repository management, you should pick up a copy of Nexus Open Source, and experiment with Hosted and Proxy repositories. You should get a sense of how Maven Settings are configured to retrieve artifacts from a single Repository Group, and you should download a copy of the free Nexus book - Repository Management with Nexus. Once you've familiarized yourself with Nexus Open Source, you can easily upgrade to Nexus Professional by downloading and installing Nexus Professional. Nexus stores all of your repository data and configuration in a directory named sonatype-work which is separate from the Nexus application directory.

...if you are looking for more stability and control

If you depend directly on public repositories such as the Maven Central repository or the various repositories maintained by organizations like Codehaus or the Apache Software Foundation, you rely on these servers to be available to your developers 100% of the time. If a public repository goes down for maintenance, so does your development process. With a local proxy of Maven artifacts, you buy yourself a stable, isolated build. Even if a public repositories becomes unavailable, you will still be able to build your software against artifacts cached in your own Nexus installation.

...if you need to manage internal software distribution

If your organization needs to support collaboration between internal teams, you can use Nexus to support the distribution of internal software. With Nexus, sharing components between internal groups is as easy as adding a dependency from Maven Central. Just publish a JAR to Nexus, configure the appropriate repositories groups and inform others in our organization of the Maven coordinates. Using a repository management doesn't just make it easier to proxy external software artifacts, it makes it easier to share internal artifacts.

...if you need an intelligent local proxy

Many developers run Nexus on a local workstation as a way to gain more control over the repositories used by Nexus. This is also a great way to start evaluating Nexus. Download and install Nexus on your local workstation and point your Maven settings at <http://localhost:8081/nexus>. When you need to add a new repository, all you need to do is change the configuration of your local Nexus installation.

...if you need to integrate with an LDAP server

If you need to integrate Nexus with an an LDAP server, download Nexus Open Source. Nexus pro-

vides documented integration with popular LDAP servers such as OpenLDAP, Microsoft's Active Directory Server, and any other directory product which implements the LDAP standard.

1.4.2 Use Nexus Professional...

...if you are looking for Professional Support

When you purchase Nexus Professional, you are purchasing one year of support from the team that created the industry-standard in repository management. With Nexus Professional, you not only get a capable repository manager, you get the peace of mind that help is just a phone call away. Sonatype also offers an array of implementation and migration services for organizations looking for an extra level of assistance. Contact Sonatype Sales for more information, call +1 (888) 866-2836.

...if you need a repository manager that can support release and quality assurance decisions:: Nexus Professional's Staging Suite can track the status of a software release and make sure that different decision makers are notified and supported during a software release. If you are looking for a repository manager that can automate and support software releases, download Nexus Professional and start learning about Staged repositories and Staging Rule-sets. When you start using Nexus Professional, your operations, quality assurance, and development teams can use the repository manager as a central point of collaboration.

...if you need more control over external artifacts

If you need more control over which external artifacts can be referenced and used in internal projects, you will need to use the Nexus Procurement Suite which is a part of Nexus Professional. While repositories like Maven Central are a great convenience, allowing your developers carte blanche access to any external library is often unacceptable in today's legal and regulatory environment. Nexus Professional's Procurement Suite allows you to enforce standards for external libraries. If you want to ensure that every dependency is evaluated for security or license compliance, download Nexus Professional.

...if you develop software for an Open Source project

Are you developing an open source project? If so, most open source projects qualify for a free Nexus Professional license. Open source projects can qualify for a free Professional license, or they can take advantage of free Nexus Professional hosting on <http://oss.sonatype.org>. Sonatype is very committed to supporting the development of quality open source and this is our way of giving back to the community.

...if you are developing and deploying to OSGi platforms

If you are developing OSGi components using OBR repositories, or if you are developing OSGi components using the P2 repository format, you will need to use the OSGi support available in the

Nexus Professional distribution. Nexus Professional supports a wider array of repository formats than Nexus Open Source. As the industry moves toward OSGi as a standard, you should be using a product which supports these emerging standards as well as the existing repository formats used by millions of developers.

... if you need to integrate with Enterprise-level Security (LDAP and Crowd):: If you need to integrate Nexus with an Atlassian Crowd server or an enterprise LDAP deployment involving multiple servers or multiple LDAP schemas, download Nexus Professional. While Nexus Open Source provides extension points for writing custom security realms, Nexus Professional provides solid LDAP and Crowd support for the large, mission-critical LDAP deployments. If you need to support LDAP fail-over and federation, use Nexus Professional.

1.4.3 Comparing Nexus Open Source and Nexus Professional Features

The following table summarizes the differences between Nexus Open Source and Nexus Professional:

	Nexus Feature	OS	Pro		Nexus Feature	OS	Pro
The Basics	Proxy Remote Repositories	✓	✓	Search by Identifier (GAV)	✓	✓	
	Host Repositories	✓	✓	Search by Checksum	✓	✓	
	Group Repositories	✓	✓	Search by Class Name	✓	✓	
	M1 to M2 Conversion	✓	✓	Search by Keyword	✓	✓	
	Checksum Validation	✓	✓	Intelligent, Later Version Results			
	Integration with m2eclipse	✓	✓	Search by Metadata	✗	✓	
	Rich AJAX UI using ExtJS	✓	✓	Intuitive Search Interface	✓	✓	
	Advanced Task Scheduling	✓	✓	Generate Nexus Index	✓	✓	
	Snapshot Cleanup	✓	✓	Download Nexus Index	✓	✓	
	On-the-fly Metadata Correction	✓	✓	Role-based Access Control	✓	✓	
Advanced	Intelligent Not Found Cache	✓	✓	LDAP/LDAPS Integration	✓	✓	
	Archetype Catalog Update	✓	✓	Multiple LDAP Servers / Mirrors	✗	✓	
	Remote Repository Browsing	✓	✓	Intelligent Caching for LDAP	✗	✓	
	Repository Auto-block/Unblock	✓	✓	Atlassian Crowd Integration	✗	✓	
	Grouping Repository Groups	✓	✓	Custom Authentication Realms	✓	✓	
	Archive Browsing	✗	✓	Self-serve User Sign Up	✗	✓	
	Viewing JavaDoc	✗	✓	Run within a Servlet Container	✓	✗	
	Extensible Plugin API	✓	✓	Run as Service on Linux	✓	✓	
	Extensible AJAX UI	✓	✓	Run as Service on Windows	✓	✓	
	Complete REST API	✓	✓	Conversion from Archiva	✓	✓	
Extensibility Types	RSS Feed of New Artifacts	✓	✓	Conversion from Artifactory	✓	✓	
	RSS Feed of System Changes	✓	✓	Configuration Backup	✗	✓	
	Maven 1 Repositories	✓	✓	Custom Metadata	✗	✓	
	Maven 2 Repositories	✓	✓	Artifact Procurement	✗	✓	
	Maven Project Sites	✗	✓	Procurement Rules	✗	✓	
	Eclipse Update Sites	✗	✓	Staged Artifact Deployment	✗	✓	
	P2 Repositories	✗	✓	Promotion RuleSets	✗	✓	
	OSGi Bundle Repositories	✗	✓	Staged Artifact Bundle Uploads	✗	✓	
	RubyGems	✓	✓	Maven Settings Management	✗	✓	

1.5 History of Nexus

Proximity in December 2005 as he was trying to find a way to isolate his own systems from an incredibly slow ADSL connection provided by a Hungarian ISP. Proximity started as a simple web application to proxy artifacts for a small organization with connectivity issues. Creating a local on-demand cache for Maven artifacts from the Maven Central repository gave an organization access to the artifacts on the Maven Central Repository, but it also made sure that these artifacts weren't downloaded over a very slow ADSL connection used by a number of developers. In 2007,

Sonatype asked Tamas to help create a similar product named Nexus. Nexus is currently considered the logical next step to Proximity. Nexus currently has an active development team, and portions of the indexing code from Nexus are also being used in m2eclipse.

Chapter 2

Repository Management

2.1 Repository Management

Repository managers serve two purposes: they act as highly configurable proxies between your organization and the public Maven repositories and they also provide an organization with a deployment destination for its own generated artifacts. Just as Source Code Management (SCM) tools are designed to manage source artifacts, Repository Managers have been designed to manage and track external dependencies and artifacts generated by your build. They 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.

2.1.1 Proxying Public Repositories

Proxying and caching a remote public repository can speed up your builds by reducing redundant downloads over the public Internet. If a developer in your organization needs to download version 2.5 of the Spring Framework and you are using Nexus, the dependencies (and the dependency's dependencies) only need to be downloaded from the remote repository once.

With a high-speed connection to the Internet this might seem like a minor concern, but if you are constantly asking your developers to download hundreds of megabytes of third-party dependencies, the real cost savings are going to be the time it takes Maven to check for new versions of dependencies and to

download dependencies over the public Internet.

Proxying and serving Maven dependencies from a local repository cache can save you hundreds of HTTP requests over the public Internet, and, in very large multi-module projects, this can shave minutes from a build.

2.1.2 Managing Releases and Snapshots

If your project is relying on a number of SNAPSHOT dependencies, Maven will need to regularly check for updated versions of these snapshots. Depending on the configuration of your remote repositories, Maven will check for SNAPSHOT updates periodically, or it might be checking for SNAPSHOT updates on every build. When Maven checks for a snapshot update it needs to interrogate the remote repository for the latest version of the SNAPSHOT dependency. Depending on your connection to the public Internet and the load on the Maven Central repository, a SNAPSHOT update can add seconds to your project's build for each SNAPSHOT dependency you rely upon.

When you host a local repository proxy with Nexus, you reduce the amount of time it takes for Maven to check for a newer version as your build interacts with a local repository cache. If you develop software with SNAPSHOT dependencies, using a local repository manager will save you a considerable amount of time, your 5-10 second SNAPSHOT update checks against the public central repository are going to execute in hundreds of milliseconds (or less) when they are executed against a local resource.

2.1.3 Getting Control of Dependencies

In addition to the simple savings in time and bandwidth, a repository manager provides an organization with control over what is downloaded by Maven. You can include or exclude specific artifacts from the public repository, and having this level of control over what is downloaded from the Maven Central repository is a prerequisite for many organizations which have a need for strict standards for the quality and security of the dependencies used in an enterprise system.

If you want to standardize on a specific version of a dependency like Hibernate or Spring you can enforce this standardization by only providing access to a specific version of an artifact in Nexus. You might be concerned with making sure that every external dependency has a license compatible with your legal standards for adopting and integrating open source libraries. If you are producing an application which is distributed, you might want to make sure that no one inadvertently adds a dependency on a third-party library covered under a copy-left license like the GPL. All of this is possible with Nexus.

Repository managers are a central point of access to external binary software artifacts and dependencies your system relies upon. Nexus provides a level of control that is essential when you are trying to track and manage the libraries and frameworks your software depends upon.

2.1.4 A Nexus for Collaboration

Aside from the benefits of mediating access to remote repositories, a repository manager also provides an important platform for collaborative software development. Unless you expect every member of your organization to download and build every single internal project from source, you will want to provide a mechanism for developers and departments to share binary artifacts (both SNAPSHOTs and releases) for internal software projects. Internal groups often consume the APIs and systems which are generated by other internal groups, when you adopt Nexus as a deployment platform for internal artifacts, you can easily share components and libraries between groups of developers.

Nexus provides you with a deployment target for your software components. Once you install Nexus, you can start using Maven to deploy snapshots and releases to internal repositories which can then be combined with other repositories in repository groups. Over time, this central deployment point for internal projects becomes the fabric for collaboration between different development teams and operations. Nexus is the secret ingredient that allows an organization to scale its development effort without sacrificing agility.

2.2 What is a Repository?

Maven developers are familiar with the concept of a repository: a collection of binary software artifacts and metadata stored in a defined directory structure which is used by clients such as Apache Ivy to retrieve binaries during a build process. In the case of the Maven repository, the primary type of binary artifact is a JAR file containing Java bytecode, but there is no limit to what type of artifact can be stored in a Maven repository. For example, one could just as easily deploy documentation archives, source archives, Flash libraries and applications, or Ruby libraries to a Maven repository. A Maven repository provides a platform for the storage, retrieval, and management of binary software artifacts and metadata.

In Maven, every software artifact 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 artifacts which a given component depends upon for successful compilation or execution.

When Maven downloads a dependency from a repository, it also downloads that dependency's POM. Given a dependency's POM, Maven can then download any other libraries which are required by that

dependency. The ability to automatically calculate a project's dependencies and transitive dependencies is made possible by the standard and structure set by the Maven repository.

Maven and other tools such as Ivy which interact with a repository to search for binary software artifacts, model the projects they manage, and retrieve software artifacts on-demand from a repository. When you download and install Maven without any customization, Maven will retrieve artifacts from a Maven Central repository which serves millions of Maven users every single day. While you can configure Maven to retrieve binary software artifacts from a collection of mirrors, the best-practice is to install Nexus and use it to proxy and cache the contents of Central on your own network.

In addition to Central, there are a number of major organizations such as Red Hat, Oracle, and Codehaus which maintain separate repositories.

While this might seem like a simple, obvious mechanism for distributing artifacts, the Java platform existed for several years before the Maven project created a formal attempt at the first repository for Java artifacts. Until the advent of the Maven repository in 2002, a project's dependencies were gathered in a manual, ad-hoc process and were often distributed with a project's source code. As applications grew more and more complex, and as software teams developed a need for more complex dependency management capabilities for larger enterprise applications, Maven's ability to automatically retrieve dependencies and model dependencies between components became an essential part of software development.

2.2.1 Release and Snapshot Repositories

A repository stores two types of artifacts: releases and snapshots. Release repositories are for stable, static release artifacts and snapshot repositories are frequently updated repositories that store binary software artifacts from projects under constant development.

While it is possible to create a repository which serves both release and snapshot artifacts, repositories are usually segmented into release or snapshot repositories serving different consumers and maintaining different standards and procedures for deploying artifacts. Much like the difference between a production network and a staging network, a release repository is considered 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 artifacts can be deployed and changed frequently without regard for stability and repeatability concerns.

The two types of artifacts managed by a repository manager are:

Release

A release artifact is an artifact which was created by a specific, versioned release. For example,

consider the 1.2.0 release of the commons-lang library stored in the Maven Central repository. This release artifact, 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 Maven Central repository. Released artifacts 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 artifact is associated with a PGP signature, an MD5 and SHA checksum which can be used to verify both the authenticity and integrity of the binary software artifact.

Snapshot

Snapshot artifacts are artifacts generated during the development of a software project. A Snapshot artifact has both a version number such as "1.3.0" or "1.3" and a timestamp in its name. For example, a snapshot artifact 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 which know how to consume snapshot artifacts from a repository also know how to interrogate the metadata associated with a Snapshot artifact to retrieve the latest version of a Snapshot dependency from a repository.

A project under active development produces SNAPSHOT artifacts that change over time. A release is comprised of artifacts which will remain unchanged over time.

2.2.2 Repository Coordinates

Repositories and tools like Maven know about a set of coordinates including the following components: 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 artifacts 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 artifact is an identifier for a software component. An artifact can represent an application or a library; for example, if you were creating a simple web application your project might have the artifactId "simple-webapp", and if you were creating a simple library, your artifact might be "simple-library". The combination of groupId and artifactId must be unique for a project.

version

The version of a project follows the established convention of Major, Minor, and Point release versions. For example, if your simple-library artifact 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 artifact it is managing. Packaging can be anything that describes any binary software format including ZIP, SWC, SWF, NAR, WAR, EAR, SAR.

2.2.3 Addressing Resources in a Repository

Tools designed to interact Maven repositories translate artifact 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 would also download 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 dependencies which would then be retrieved from the same repository using the same URL patterns.

2.2.4 The Maven Central Repository

The most useful Maven repository is the Maven Central repository. The Maven Central repository contains almost 90,000 software artifacts occupying around 70 GB of disk space. You can look at Central 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 Maven Central repository:

Artifact Metadata

All software artifacts added to Central require proper metadata including a Project Object Model

(POM) for each artifact which describes the artifact itself, and any dependencies that software artifact might have.

Release Stability

Once published to the Maven Central repository, an artifact and the metadata describing that artifact never change. This property of release repositories guarantees that projects which depend on releases will be repeatable and stable over time. While new software artifacts are being published to central every day, once an artifact is assigned a release number on Central, there is a strict policy against modifying the contents of a software artifact after a release.

Repository Mirrors

Central is a public resource, and it is currently used by the millions of developers who have adopted Maven and the tools that understand how to interact with the Maven repository structure. There are a series of mirrors for the Central repository which are constantly synchronized with Central. Users are encouraged to query central for project metadata and cryptographic hashes and they are encouraged to retrieve the actual software artifacts from one of Central's many mirrors. Tools like Nexus are designed to retrieve metadata from Central and artifact binaries from mirrors.

Artifact Security

The Maven Central repository contains cryptographic hash cryptographic hashes and PGP signatures which can be used to verify the authenticity and integrity of software artifacts served from Central or one of the many mirrors of Central.

2.3 What is a Repository Manager

If you use Maven, you are using a repository to retrieve artifacts and Maven plugins. In fact, Maven used a Maven repository to retrieve core plugins that implement the bulk of the features used in your builds. 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 the Maven Central 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 Central into your organization, you can use it to proxy Central, and host your own repositories for internal and external use. In this section, we discuss the core functionality which defines what a repository manager does.

Put simply, a repository manager provides two core features:

- The ability to proxy a remote repository and cache artifacts saving both bandwidth and time required to retrieve a software artifact from a remote repository, and
 - The ability the host a repository providing an organization with a deployment target for software artifacts.
-

In addition to these two core features, a repository manager also allows you to manage binary software artifacts through the software development life-cycle, search and catalogue software artifacts, audit development and release transactions, and integrate with external security systems such as LDAP. The following sections define the feature sets of Nexus Open Source and Nexus Professional.

2.3.1 Core Capabilities of a Repository Manager

The base-line features of a repository manager are a description of the core capabilities of Nexus Open Source. Nexus Open Source provides for the:

Management of Software Artifacts

A repository manager is able to manage packaged binary software artifacts. In Java development, this would include JARs containing bytecode, source, or javadoc. In other environments, such as Flex, this would include any SWCs or SWFs generated by a Flex build.

Management of Software Metadata

A repository manager should have some knowledge of the metadata which describes artifacts. In a Maven repository this would include project coordinates (groupId, artifactId, version, classifier) and information about a given artifact's releases.

Proxying of External Repositories

Proxying an external repository yields more stable builds as the artifacts used in a build can be served to clients from the repository manager's cache even if the external repository becomes unavailable. Proxying also saves bandwidth and time as checking for the presence of an artifact on a local network is often orders of magnitude faster than querying a heavily loaded public repository.

Deployment to Hosted Repositories

Organizations which deploy internal snapshots and releases to hosted repositories have an easier time distributing software artifacts across different teams and departments. When a department or development group deploys artifacts to a hosted repository, other departments and development groups can develop systems in parallel, relying upon dependencies served from both release and snapshot repositories.

Searching an Index of Artifacts

When you collect software artifacts and metadata in a repository manager, you gain the ability to create indexes and allow users and systems to search for artifacts. With the Nexus index, an IDE such as Eclipse has almost instantaneous access to the contents of all proxy repositories (including the Central repository) as well as access to your own internal and 3rd party artifacts. While the Central repository transformed the way that software is distributed, the Nexus index format brings the power of search to massive libraries of software artifacts.

Infrastructure for Artifact Management

A repository manager should also provide the appropriate infrastructure for managing software

artifacts and a solid API for extension. In Nexus, Sonatype has provided a plugin API which allows developers to customize both the behaviour, appearance, and functionality of the tool.

2.3.2 Additional Features of a Repository Manager

Once you adopt the core features of a repository manager, you start to view a repository manager as a tool which enables more efficient collaboration between development groups. Nexus Professional builds upon the foundations of a repository manager and adds capabilities such as Procurement and Staging.

Managing Project Dependencies

Many organizations require some level of oversight over the open source libraries and external artifacts that are let into an organization's development cycle. An organization could have specific legal or regulatory constraints which requires every dependency to be subjected to a rigorous legal or security audit before it is integrated into a development environment. Another organization might have an architecture group which needs to make sure that a large set of developers only has access to a well-defined list of dependencies or specific versions of dependencies. Using the Procurement features of Nexus Professional, managers and architecture groups have the ability to allow and deny specific artifacts from external repositories.

Managing a Software Release

Nexus Professional adds some essential work-flow to the process of staging software to a release repository. Using Nexus Professional, developers can deploy to a staging directory which can trigger a message to a Release Manager or to someone responsible for QA. Quality assurance (or a development manager) can then test and certify a release having the option to promote a release to the release repository or to discard a release if it didn't meet release standards. Nexus Professional's staging features allow managers to specify which personnel are allowed to certify that a release can be promoted to a release repository giving an organization more control over what software artifacts are released and who can release them.

Integration with LDAP

Nexus integrates with an LDAP directory, allowing an organization to connect Nexus to an existing directory of users and groups. Nexus authenticates users against an LDAP server and provides several mechanisms for mapping existing LDAP groups to Nexus roles.

Advanced Security

Using Nexus Professional, an organization can define a master User Password Encryption Key. Each user will be given a separate Maven settings file with an encrypted password using the Maven Nexus plugin. When users interact with Nexus, Nexus uses the User Password Encryption Key to decrypt a user's Nexus credentials avoiding the need to send an easily compromised plain-text password over the network.

Settings Templates

Nexus Professional allows you to define Maven settings templates for developers. Developers can

then automatically receive updates to Maven settings (`~/.m2/settings.xml`) using the Maven Nexus plugin. The ability to define Maven settings templates and to distribute customized Maven settings files to developers makes it easy for an organization to change global profiles or repository configuration without relying on developers to manually install a new settings file in a development environment.

Support for Multiple Repository Formats

Nexus Professional supports the P2 and the OSGi Bundle repository format used by the new Eclipse provisioning platform and OSGi developers. You can use the P2 plugin to consolidate, provision, and control the plugins that are being used in an Eclipse IDE. Using Nexus procurement, repository groups, and proxy repositories to consolidate multiple plugin repositories, an organization can use Nexus Professional to standardize the configuration of Eclipse IDE development environments.

2.4 Reasons to Use a Repository Manager

Here are a few reasons why using a repository manager is an imperative. While most people wouldn't even think of developing software without the use of a source code control system like Subversion or Perforce, the concept of using a repository manager is still something that needs development. There are many who use Maven for years without realizing the benefits of using a repository manager. This section was written as an attempt to capture some of the benefits of using a repository manager.

2.4.1 Speed Up Your Builds

When you run your multi-module project in Maven, how do you think Maven knows if it needs to update plugins or snapshot dependencies? It has to make a request for each artifact it needs to test. Even if nothing has changed, if your project depends on a few SNAPSHOTs or if you don't specify plugin version, Maven might have to make tens to hundreds of requests to a remote repository. All of these requests over the public Internet add up to real, wasted, time. I've seen complex builds cut build time by 75% after installing a local instance of Nexus. You are wasting time better spent coding waiting for your build to needlessly interrogate a remote Maven repository.

2.4.2 Save Bandwidth

The larger the organization, the more critical bandwidth savings can be. If you have thousands of developers regularly wasting good bandwidth to download the same files over and over again, using a repository

manager to keep a local cache is going to save you a good deal of bandwidth. Even for smaller organizations with limited budgets for connectivity and IT operations, having to deal with a set of developers maxing out your connection to the Internet to download the same things over and over again seems backwards.

2.4.3 Ease the Burden on Central

Running the Maven Central repository is no short order. It ain't cheap to serve the millions of requests and Terabytes of data required to satisfy the global demand for software artifacts from the Maven Central repository. Something as simple as installing a repository manager at every organization that uses Maven would likely cut the bandwidth requirements for Central by at least half. If you have more than a couple developers using Maven, install a repository manager for the sake of keeping Central available and in business.

2.4.4 Gain Predictability and Scalability

How often in the past few years has your business come to a crashing halt because of an outage? Depending on Central for your day to day operations also means that you depend on having Internet connectivity (and on the fact the Central will remain available 24/7). While Sonatype is confident in its ability to keep Central running 24/7, you should take some steps of your own to make sure that your development team isn't going to be surprised by some network outage on either end. If you have a local repository manager, like Nexus, you can be sure that your builds will continue to work even if you lose connectivity.

2.4.5 Control and Audit Dependencies and Releases

So, you've moved over to Maven (or maybe Ivy, Ivy reads the same repository), and you now have a whole room full of developers who feel empowered to add or remove dependencies and experiment with new frameworks. We've all seen this. We've all worked in places with a developer who might be more interested in experimenting than in working. It is unfortunate to say so, but there are often times when an architect, or an architecture group needs to establish some baseline standards which are going to be used in an organization. Nexus provides this level of control. If you need more oversight over the artifacts that are making it into your organization, take a look at Nexus. Without a repository manager, you are going to have little control over what dependencies are going to be used by your development team.

2.4.6 Deploy 3rd Party Artifacts

How do you deal with that one-off JAR from a vendor that is not open source, and not available on the Maven Central repository? You need to deploy these artifacts to a repository and configure your Maven instance to read from that repository. Instead of hand-crafting some POMs, download Nexus and take the two or three minutes it is going to take to get your hands on a tool that can create such a repository from 3rd-party artifacts. Nexus provides an intuitive upload form that you can use to upload any random free-floating JAR that finds its way into your project's dependencies.

2.4.7 Collaborate with Internal Repositories

Many organizations require every developer to checkout and build the entire system from source simply because they have no good way of sharing internal JARs from a build. You can solve a problem like this by splitting projects up and using Nexus as an internal repository to host internal dependencies.

For example, consider a company that has 30 developers split into three groups of 10, each group focused on a different part of the system. Without an easy way to share internal dependencies, a group like this is forced either to create an ad hoc file-system-based repository or to build the system in its entirety so that dependencies are installed in every developer's local repository.

The alternative is to separate the projects into different modules that all have dependencies on artifacts hosted by an internal Nexus repository. Once you've done this, groups can collaborate by exchanging compiled snapshot and release artifacts via Nexus. In other words, you don't need to ask every developer to checkout a massive multi-module project that includes the entire organization's code. Each group within the organization can deploy snapshots and artifacts to a local Nexus instance, and each group can maintain a project structure which includes only the projects it is responsible for.

2.4.8 Distribute with Public Repositories

If you are an open source project, or if you release software to the public, Nexus can be the tool you use to serve artifacts to external users. Think about it this way... When was the last time you cut a release for your software project? Assuming it wasn't deployed to a Maven repository, you likely had to write some scripts to package the contents of the release, maybe someone special had to sign the release with a super-secret cryptographic key. Then, you had to upload it to some web server, and then make sure that the pages that describe the upload were themselves updated. Lots of needless complexity...

If you were using something like Nexus, which can be configured to expose a hosted repository to the

outside world, you could use the packaging and assembly capabilities of Maven and the structure of the Maven repository to make a release that is more easily consumed. And, this isn't just for JAR files and Java web applications; Maven repositories can host any kind of artifact. Nexus, and Maven repositories in general, define a known structure for releases. If you are writing some Java library, publishing it to your own Nexus instance serving a public repository will make it easier for people to start using your code right away.

2.5 Adopting a Repository Manager

This section talks about the stages of moving to a repository manager. Adopting a repository manager is not an all or nothing proposition, and there are various levels (or stages) of adoption that can be distinguished when approaching repository management. On one end of the adoption spectrum is the organization that installs a repository manager just to control and consolidate access to a set of remote repositories. On the other end of the spectrum is the organization which has integrated the repository manager into an efficient software development life-cycle, using it to facilitate decision points in the life-cycle, encouraging more efficient collaboration throughout the enterprise, and keeping detailed records to increase visibility into the software development process.

2.5.1 Stage Zero: Before Using a Repository Manager

While this isn't a stage of adoption, Stage Zero is a description of the way software builds work in the absence of a repository manager. When a developer decides that he needs a particular open source software component, he will download it from the component's web site, read the documentation, and find the additional software that his components rely on (referred to as "dependencies"). Once he has manually assembled a collection of dependencies from various open source project web sites and proprietary vendors, he will place all these components somewhere on the network so that he, his team members, the build script, the QA team, and the production support team can find it. At any time, other developers may bring in other components, sometimes with overlapping dependencies, placing them in different network locations. The instructions to bring all of these ad-hoc, developer-managed components libraries together in a software build process can become very complicated and hard to maintain.

Maven was introduced to improve this build process by introducing the concept of structured repositories from which the build scripts can retrieve the software components. In Maven language, these software components or dependencies are referred to as "artifacts", a term which can refer to any generic software artifact including components, libraries, frameworks, containers, etc. Maven can identify artifacts in repositories, understand their dependencies, retrieve all that are needed for a successful build, and deploy its output back to repositories when done.

Developers using Maven without a repository manager find most of their software artifacts and dependencies in Maven Central. If they happen to use another remote repository or if they need to add a custom artifact, the solution, in Stage Zero, is to manually manipulate the files in a local repository and share this local repository with multiple developers. While this approach may yield a working build for a small team, managing a shared local repository doesn't allow an organization to scale a development effort. There is no inherent control over who can set up a local repository, who can add to them or change or delete from them, nor are there tools to protect the integrity of these repositories.

That is, until Repository Managers were introduced.

2.5.2 Stage One: Proxying Remote Repositories

This is the easiest stage to understand both in terms of benefits to an organization and action required to complete this stage. All you need to do to start Procyon a remote repository is to deploy Nexus and start the server with the default configuration. Configure your Maven clients to read from the Nexus public repository group, and Nexus will automatically retrieve artifacts from remote repositories, such as Maven Central, caching them locally.

Without a repository manager, your organization might have hundreds of developers independently downloading the same artifacts from public, remote repositories. With a repository manager, these artifacts can be downloaded once and stored locally. After Stage One, your builds run considerably faster than they did when you relied upon the Maven Central repository.

Once you've installed Nexus and you've configured all of your organization's clients to use it as a single point of access to remote repositories, you begin to realize that it now provides you with a central configuration point for the artifacts used throughout your organization. Once you've started to proxy, you can start to think about using Nexus as a tool to control policy and what dependencies are allowed to be used in your organization. Nexus Professional provides a procurement plugin which allows for fine-grained control over which artifacts can be accessed from a remote repository. This procurement feature is described in more detail in the section which deals with Life-cycle Integration.

2.5.3 Stage Two: Hosting a Repository Manager

Once you have started to proxy remote repositories and you are using Nexus as a single, consolidated access point for remote repositories, you can start to deploy your own artifacts to Nexus hosted repositories. Most people approach repository management to find a solution for proxying remote repositories, and while proxying is the most obvious and immediate benefit of installing a repository manager, hosting internally generated artifacts tends to be the stage that has the most impact on collaboration within an

organization.

To understand the benefits of hosting an internal repository, you have to understand the concept of managing binary software artifacts. Software development teams are very familiar with the idea of a source code repository or a source code management tool. Version control systems such as Subversion, Clearcase, Git, and CVS provide solid tools for managing the various source artifacts that comprise a complex enterprise application, and developers are comfortable checking source out from source control to build enterprise applications. However, past a certain point in the software development life-cycle, source artifacts are no longer relevant. A QA department trying to test an application or an Operations team attempting to deploy an application to a production network no longer needs access to the source artifacts. QA and Operations are more interested in the compiled end-product of the software development life-cycle: the binary software artifacts. A repository manager allows you to version, store, search, archive, and release binary software artifacts derived from the source artifacts stored in a source control system. A repository manager allows you to apply the same systematic operations on binary software artifacts which you currently apply to your source code.

When your build system starts to deploy artifacts to an internal repository, it changes the way that developers and development groups can interact with one another in an enterprise. Developers in one development group can code and release a stable version of an internal library, deploy this library to an internal Nexus release repository, and so share this binary artifact with another group or department. Without a repository manager managing internal artifacts, you have ad-hoc solutions and the organizational equivalent of "duct tape". How does the infrastructure group send a new library to the applications group without Nexus? Someone copies a file to a shared directory, and sends an email to the team lead. Organizations without repository managers are full of these ad-hoc processes that get in the way of efficient development and deployment.

With a repository manager, every developer and every development group within the enterprise understands and interacts with a common collaborative structure: the repository manager. Do you need to interact with the Commerce team's new API? Just add a dependency to your project and Maven will retrieve the library from Nexus automatically.

One of the other direct benefits of deploying your own artifacts to a repository such as Nexus is the ability to quickly search the metadata and contents of those artifacts both via a web UI and through IDE integration tools such as m2eclipse. When you start to deploy internal artifacts you can synchronize all development groups to a common version and naming standard, and you can use the highly configurable authentication and role-based access controls to control which developers and which development groups can deploy artifacts to specific repositories or paths within a repository.

2.5.4 Stage Three: Continuous Collaboration

Developing this collaborative model further, if your application is being continuously built and deployed using a tool like Hudson, a developer can checkout a specific module from a large multi-module build and not have to constantly deal with the entire source tree at any given time. This allows a software development effort to scale efficiently. If every developer working on a complex enterprise application needs to checkout the entire source tree every time he or she needs to make a simple change to a small component, you are quickly going to find that building the entire application becomes a burdensome bottleneck to progress. The larger your enterprise grows, the more complex your application becomes, the larger the collective burden of wasted time and missed opportunities. A slow enterprise build prevents the quick turnaround or quick feedback loop that helps your developers maintain focus during a development cycle.

Once you are building with Maven, sharing binary artifacts with Nexus, continuously testing and deploying with Hudson, and generating reports and metrics with tools like Sonar, your entire organization gains a collaborative "central nervous system" that enables a more agile approach to software development.

2.5.5 Stage Four: Life-cycle Integration

Once you've configured a repository manager to proxy remote repositories and you are using a repository manager as an integration point between developers and departments, you start to think about the various ways your repository manager can be used to support the decisions that go into software development. You can start using the repository manager to stage releases and supporting the work-flow associated with a managed release, and you can use the procurement features of a tool like Nexus Professional to give management more visibility into the origins, characteristics and open source licenses of the artifacts used during the creation of an enterprise application.

Nexus Professional enables organizations to integrate the management of software artifacts tightly with the software development life-cycle: Provisioning, Compliance, Procurement, Enterprise Security, Staging and other capabilities that support the work-flow that surrounds a modern software development effort.

Using Nexus Professional's Maven Settings management feature and integrated security features you can configure a developer's Maven settings by running a single, convenient Maven goal and downloading customized settings for a particular developer. When you use Maven and Nexus Professional together, developers can get up and running quickly, collaborating on projects that share common conventions without having to manually install dependencies in local repositories.

Provisioning

Using Nexus as an integration point between Engineering and Operations means that Engineering

can be responsible for delivering solid, tested artifacts to Quality Assurance and Operations via a standard repository format. Often development teams are roped into the production deployment story and become responsible for building entire production environments within a build system. This conflates software engineering with system administration and blurs the line between Engineering and Operations. If you use Nexus as an end-point for releases from Engineering, Operations can then retrieve, assemble, and configure an application from tested components in the Nexus repository.

Compliance

Procurement, staging, and audit logs are all features which increase the visibility into who and what is involved with your software development effort. Using Nexus Professional, Engineering can create the reports and documents which can be used to facilitate discussions about oversight. Organizations subject to various regulations often need to produce a list of components involved in a software release. Legal departments often require a list of open source licenses being used in a particular software component, and managers often lack critical visibility into the software development process.

Procurement

The ease with which today's developer can add a dependency on a new open source library and download this library from a Central repository has a downside. Organizations large and small are constantly wondering what open source libraries are being used in applications, and whether these libraries have acceptable open source licenses for distribution. The Procurement features of Nexus Professional give architects and management more oversight over the artifacts which are allowed into an organization. Using the Procurement features, a Nexus administrator or Procurement manager can allow or deny specific artifacts by group, version, or path. You can use the procurement manager as a firewall between your own organization's development environment and the 95,000 artifacts available on the Maven Central repository.

Enterprise Security

Nexus' LDAP integration allows an enterprise to map existing LDAP groups to Nexus roles and provides Nexus administrators with a highly configurable interface to control which individuals or groups have access to a fine-grained set of Nexus permissions.

Staging

Nexus Professional adds an important step to the software release work-flow, adding the concept of a managed (or staged) release to a hosted repository. When a developer needs to perform a production release, Nexus Professional can isolate the artifacts involved in a release in a staged repository which can then be certified and tested. A manager or a quality assurance tester can then promote or discard a release. The staging feature allows you to specify the individuals that are allowed to promote a release and keeps an audit of who was responsible for testing, promoting, or discarding a software release.

Chapter 3

Installing and Running Nexus

3.1 Nexus Prerequisites

Nexus Open Source and Nexus Professional only have one prerequisite, a Java Runtime Environment (JRE) compatible with Java 5 or higher. Nexus is most often run with the JRE that is bundled with a Java Development Kit (JDK) installation, and it can be run with the latest version of Oracle's JDK for Java 6 or Java 7. To download the latest release of the Oracle JDK, go to <http://www.oracle.com/technetwork/java/javase/downloads/index.html>, and download the latest Java 6 or Java 7 JDK.

3.2 Downloading Nexus

There are two distributions of Nexus: [Nexus Open Source](#) and [Nexus Professional](#). Nexus Open Source is a fully-featured repository manager which can be freely used, customized, and distributed under the Eclipse Public License (EPL) Version 1. Nexus Professional is a distribution of Nexus with features that are relevant to large enterprises and organizations which require complex procurement and staging workflows in addition to more advanced LDAP integration, Atlassian Crowd support, and other development infrastructure. The differences between Nexus Open Source and Nexus Professional are explored in the previous chapter.

3.2.1 Downloading Nexus Open Source

To download Nexus Open Source go to <http://nexus.sonatype.org/nexus/go> and download the latest Nexus Open Source distribution by clicking on the appropriate button for a ZIP or a Gzip TAR archive (TGZ) shown in Figure 3.1. Your download will be file named nexus-2.0.6-bundle.zip or nexus-2.0.6-bundle.tar.gz

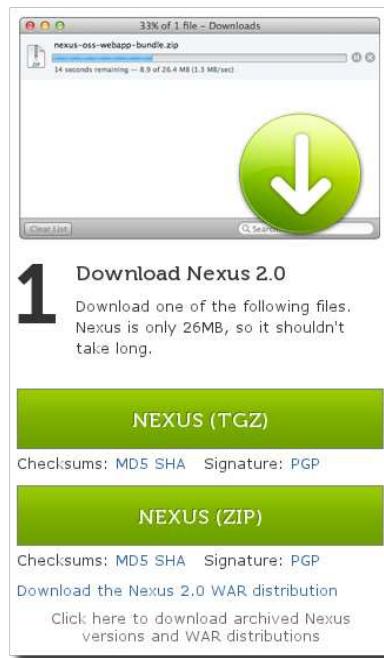


Figure 3.1: Downloading Nexus Open Source

Older versions can be downloaded following the link at the bottom of Figure 3.1 and selecting a version and archive type in the page displayed in Figure 3.2.



Figure 3.2: Selecting the Nexus Open Source Distribution Download

Nexus Open Source can also be deployed as a web application in a servlet container like Jetty or Tomcat or an application server like Glassfish or JBoss. Instructions for installing Nexus as a WAR are found in Section 3.9.

3.2.2 Downloading Nexus Professional

To download Nexus Professional, go to the [Nexus Trial evaluation form](#).

Fill out the form and then check your email account for download instructions. After filling out a brief contact form and agreeing to the evaluation license, you will be sent an email with a link to download the Nexus Professional distribution. Nexus is available as archive files `nexus-professional-2.0.6-bundle.zip` and `nexus-professional-2.0.6-bundle.tar.gz`.

3.3 Installing Nexus

The following instructions are for installing Nexus Open Source or Nexus Professional as a stand-alone server. Nexus comes bundled with a Jetty instance which listens to all configured IP addresses on a host (0.0.0.0) and runs on port 8081 by default. If you would like to run Nexus as a web application in an existing application server or servlet container, please refer to the instructions in Section 3.9.

Installing Nexus is straightforward. Unpack the Nexus web application archive in a directory. If you are installing Nexus on a local workstation to give it a test run, you can install it in your home directory or wherever you like; Nexus doesn't have any hard coded directories, it will run from any directory. If you downloaded the ZIP

```
$ unzip nexus-2.0.6-bundle.zip
```

And, if you download the GZip'd TAR archive, run:

```
$ tar xvzf nexus-2.0.6-bundle.tar.gz
```

For Nexus professional the equivalent commands would be

```
$ unzip nexus-professional-2.0.6-bundle.zip  
$ tar xvzf nexus-professional-2.0.6-bundle.tar.gz
```

Note

There are some known incompatibilities with the version of tar provided by Solaris and the gzip tar format. If you are installing Nexus on Solaris, you must use the GNU tar application, or you will end up with corrupted files.

Note

If you are installing Nexus on a server, you might want to use a directory other than your home directory. On a Unix machine, this book assumes that Nexus is installed in /usr/local/nexus-2.0.6 with a symbolic link /usr/local/nexus to the nexus directory. Using a generic symbolic link nexus to a specific version is a common practice which makes it easier to upgrade when a newer version of Nexus is made available.

```
$ sudo cp nexus-2.0.6-bundle.tar.gz /usr/local  
$ cd /usr/local  
$ sudo tar xvzf nexus-2.0.6-bundle.tar.gz  
$ ln -s nexus-2.0.6 nexus
```

Although it isn't required for Nexus to run, you may want to set an environment variable NEXUS_HOME in your environment which points to the installation directory of Nexus. This chapter will refer to this location as *\$NEXUS_HOME*

Note

On Windows you should install Nexus outside Program Files to avoid problems with Windows file registry virtualization. If you plan to run Nexus as a specific user you could install into the AppData/Local directory of that users home directory. Otherwise simply go with e.g. C:\nexus or something similar.

The Nexus installation directory nexus-2.0.6 or nexus-professional-2.0.6 has a sibling directory named sonatype-work. This directory contains all of the repository and configuration data for Nexus and is stored outside of the Nexus installation directory to make it easier to upgrade to a newer version of Nexus.

By default, this directory is always a sibling to the nexus installation directory; if you installed nexus in the /usr/local directory it would also contain a sonatype-work sub-directory with a nested nexus directory containing all of the content and configuration. The location of the sonatype-work directory can be customized by altering the nexus-work property in *\$NEXUS_HOME/conf/nexus.properties*

3.4 Upgrading Nexus

Since Nexus separates its configuration and data storage from the application, it is easy to upgrade an existing Nexus installation.

To upgrade Nexus, unpack the Nexus archive in the directory which contains the existing Nexus installation. Once the archive is unpacked, the new Nexus application directory should be a sibling to your existing sonatype-work/ directory.

If you have defined a symbolic link for the version of Nexus to use, stop the server and change that to point at the new Nexus application directory. When you start the new instance of Nexus it will read the existing repository configuration from the sonatype-work directory. Depending on the version you upgrade from and to some maintenance tasks like rebuilding the internal indices can be necessary. Please refer to the upgrade notes of the new release for more information on this.

If you are using any additional plugins supplied by Sonatype, the new version of Nexus you downloaded will contain a newer version of the plugin. Be sure to copy the new version from the optional-plugins folder to the plugin-repository folder as documented in Section [3.12](#) and restart Nexus.

Externally supplied plugins are updated by simply replacing the folder with the plugin with the new version.

Note

The same upgrade process can be used to change from the open source to the professional version of Nexus.

3.5 Running Nexus

When you start Nexus, you are starting a web server on the default port of 0.0.0.0:8081. Nexus runs within a servlet container called Jetty and it is started with a native service wrapper called the [Tanuki Java Service Wrapper](#). This service wrapper can be configured to run Nexus as a Windows service or a Unix daemon. Nexus ships with generic startup scripts for Unix-like platforms called nexus and for Windows platforms called nexus.bat in the \$NEXUS_HOME/bin folder. To start Nexus on a Unix-like platform like Linux, MacOSX or Solaris use

```
cd /usr/local/nexus  
./bin/nexus console
```

Similarly starting on Windows can be done with the nexus.bat file. Starting Nexus with the console command will leave Nexus running in the current shell and display the log output right there.

On Unix system you can start Nexus detached from the starting shell with will the start command even when not yet installed as a service.

```
./bin/nexus start
```

When executed you should see a feedback message and can then follow the start-up process viewing the log file logs/wrapper.log changes.

```
Starting Nexus Repository Manager...  
Started Nexus Repository Manager.  
$ tail -f logs/wrapper.log
```

At this point, Nexus will be running and listening on all IP addresses (0.0.0.0) that are configured for the current host on port 8081. To use Nexus, fire up a web browser and type in the URL <http://localhost:8081/nexus> and you should see the Nexus user interface as displayed in Figure 3.3

While we use "localhost" throughout this book, you may need to use the IP Loopback Address of "127.0.0.1" or the IP address assigned to the machine running Nexus. Click on the "Log In" link in the upper right-hand corner of the web page, and you should see the login dialog displayed in Figure 3.4.

Tip

The default administrator username and password combination is "admin" and "admin123".



Figure 3.3: Nexus Application Window



Figure 3.4: Nexus Login Dialog (default login/password is admin/admin123)

The files from Java Service Wrapper used for the start up process can be found in \$NEXUS_HOME/bin/jsw and are separated into generic files like the wrapper.conf configuration file in conf and a number of libraries in lib. An optional wrapper.conf include allows you to place further configuration optionally in \$NEXUS_HOME/conf/wrapper-override.conf.

The platform specific directories are available for backwards compatibility with older versions only and should not be used. A full list of directories follows:

```
$ cd /usr/local/nexus/jsw  
$ ls -1  
conf
```

```
lib  
license  
linux-ppc-64  
linux-x86-32  
linux-x86-64  
macosx-universal-32  
macosx-universal-64  
solaris-sparc-32  
solaris-sparc-64  
solaris-x86-32  
windows-x86-32  
windows-x86-64
```

Tip

The start-up script `nexus` supports the common service commands `stop`, `start`, `restart`, `status`, `console` and `dump`.

3.6 Post-Install Checklist

Nexus ships with some default passwords and settings for repository indexing that need to be changed for your installation to be useful (and secure). After installing and running Nexus, you need to make sure that you complete the following tasks:

3.6.1 Step 1: Change the Administrative Password and Email Address

The administrative password defaults to `admin123`. The first thing you should do to your new Nexus installation is change this password. To change the administrative password login as "admin" with the password "admin123", and click on Change Password under the Security menu in the left-hand side of the browser window. For more detailed instructions, see Section [5.8](#).

3.6.2 Step 2: Configure the SMTP Settings

Nexus can send user-name and password recovery emails, to enable this feature, you will need to configure Nexus with a SMTP Host and Port as well as any necessary authentication parameters that Nexus needs

to connect to the mail server. To configure the SMTP settings, load the server configuration dialog shown in Figure 6.2.

3.6.3 Step 3: Enable Remote Index Downloads

Nexus ships with three important proxy repositories for the Maven Central repository, Apache Snapshot repository, and the Codehaus Snapshot repository. Each of these repositories contains thousands (or tens of thousands) of artifacts and it would be impractical to download the entire contents of each. To that end, most repositories maintain an index which catalogues the entire contents and provides for fast and efficient searching. Nexus uses these remote indexes to search for artifacts, but we've disabled the index download as a default setting. To download remote indexes:

1. Click on Repositories under the Administration menu in the left-hand side of the browser window.
2. Select each of the three proxy repositories and change Download Remote Indexes to true in the Configuration tab. You'll need to load the dialog shown in Figure 6.13 for each of the three repositories.

This will trigger Nexus to re-index these repositories, during which the remote index files will be downloaded. It might take Nexus a few minutes to download the entire index, but once you have it, you'll be able to search the entire contents of the Maven repository.

Once you've enabled remote index downloads, you still will not be able to browse the complete contents of a remote repository. Downloading the remote index allows you to search for artifacts in a repository, but until you download those artifacts from the remote repository they will not show in the repository tree when you are browsing a repository. When browsing a repository, you will only be shown artifacts which have been downloaded from the remote repository.

3.6.4 Step 4: Change the Deployment Password

The deployment user's password defaults to deployment123. Change this password to make sure that only authorized developers can deploy artifacts to your Nexus installation. To change the deployment password: log in as an administrator, click on Security to expand the Security menu, then click on Users. You should then see a list of users. Right-click on the deployment user and select "Set Password".

3.6.5 Step 5: Install Professional License (Nexus Professional Only)

If you are running Nexus Professional and you have purchased a license from Sonatype, you will need to upload your license. This license will be emailed to you after you have purchased Nexus Professional from <http://store.sonatype.com>. To load the License tab, click on Licensing under the Enterprise menu in the Nexus application menu. For more information about installing a Nexus Professional license file, see Section 3.10.

3.6.6 Step 6: If necessary, set the LANG Environment Variable

If your Nexus instance needs to store configuration and data using an international character set, you should set the LANG environment variable. The Java Runtime will adapt to the value of the LANG environment variable and ensure that configuration data is saved using the appropriate character type. If you are starting Nexus as a service, place this environment variable in the start-up script found in /etc/init.d/nexus. For more information about locale settings in Ubuntu read <https://help.ubuntu.com/community/Locale>

3.7 Configuring Nexus as a Service

When installing Nexus for production usage you should configure Nexus as a service, so it starts back up after server reboots. It is good practice to run that service or daemon as a specific user that has only the required access rights. The following sections provide instructions for configuring Nexus as a service or daemon on various operating systems.

3.7.1 Running as a Service on Linux

You can configure Nexus to start automatically, by copying the nexus script to the /etc/init.d directory. On a Linux system perform the following operations as the root user:

1. Copy either \$NEXUS_HOME/bin/nexus to /etc/init.d/nexus
2. Make the /etc/init.d/nexus script executable - chmod 755 /etc/init.d/nexus
3. Edit this script changing the following variables:

- a. Change NEXUS_HOME to the absolute folder location e.g. NEXUS_HOME="/usr/local/nexus"
 - b. Optionally change PIDDIR to /var/run
 - c. If Java is not on the default path for the user running Nexus, add a JAVA_HOME variable which points to your local Java installation and add a \$JAVA_HOME/bin to the PATH
4. (Optional) Set the RUN_AS_USER to "nexus". If you do this, you will need to:
- a. Create a nexus user
 - b. Change the Owner and Group of your nexus install directory to nexus

Note

If you set the "RUN_AS_USER" variable, you'll have to change the "pid" directory to point to a directory where this user has read/write permissions. In most Linux distributions, /var/run is only writable by root. The properties that would need to be added to customize the PID file location is "wrapper.pid". For more information about this property and how it would be configured in wrapper.conf, see: <http://wrapper.tanukisoftware.com/doc/english/properties.html>

3.7.1.1 Add Nexus as a Service on Red Hat, Fedora, and CentOS

This script has the appropriate chkconfig directives, so all you need to do to add Nexus as a service is run the following commands:

```
$ cd /etc/init.d  
$ chkconfig --add nexus  
$ chkconfig --levels 345 nexus on  
$ service nexus start  
Starting Sonatype Nexus...  
$ tail -f /usr/local/nexus/logs/wrapper.log
```

The second command adds nexus as a service to be started and stopped with the service command and managed by the chkconfig manager. 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. The service command starts Nexus, and the last command tails the wrapper.log to verify that Nexus has been started successfully. If Nexus has started successfully, you should see a message notifying you that Nexus is listening for HTTP.

3.7.1.2 Add Nexus as a Service on Ubuntu and Debian

The process for setting Nexus up as a service on Ubuntu differs slightly from the process used on a Red Hat variant. Instead of running chkconfig, you should run the following sequence of commands once you've configured the start-up script in /etc/init.d

```
$ cd /etc/init.d  
$ update-rc.d nexus defaults  
$ service nexus start  
Starting Sonatype Nexus...  
$ tail -f /usr/local/nexus/logs/wrapper.log
```

3.7.2 Running as a Service on Mac OS X

The standard way to run a service on Mac OS X is by using launchd, which uses plist files for configuration. An example plist file for Nexus is shown [A sample com.sonatype.nexus.plist file](#).

A sample com.sonatype.nexus.plist file

```
<?xml version="1.0" encoding="UTF-8"?>  
<!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>/usr/local/nexus/bin/nexus</string>  
<string>console</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
```

Tip

Consider setting up a different user to run Nexus and adapt permissions and the RUN_AS_USER setting in the nexus startup script.

With this setup Nexus will start 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
```

3.7.3 Running as a Service on Windows

The start-up script for Nexus on Windows platforms is bin/nexus.bat. Besides the standard commands for starting and stopping the service it has the additional commands install and uninstall. Running these commands with elevated privileges will set up the service for you or remove it as desired. Once installed as a service with the install command the batch file can be used to start and stop the service. In addition the service will be available in the usual Windows service management console with the nexus.

3.8 Running Nexus Behind a Proxy

If you installed Nexus as a stand-alone application, Nexus is running on a high-performance servlet container based on Java NIO. From a performance perspective, there is no reason for you not to run Nexus by itself without a proxy. Yet, more often than not, organizations run applications behind a proxy for security concerns and to consolidate multiple disparate applications using tools like mod_rewrite. For this reason, we've included some brief instructions for configuration Apache httpd. We assume that you've already installed Apache 2, and that you are using a Virtual Host for www.somecompany.com.

Let's assume that you wanted to host Nexus behind Apache HTTPD at the URL <http://www.somecompany.com>. To do this, you'll need to change the context path that Nexus is served from.

1. Edit nexus.properties in `$NEXUS_HOME/conf`. You'll see an element named `nexus-webapp-context-path`. Change this value from `/nexus` to `/`
 2. Restart Nexus and Verify that it is available on <http://localhost:8081/>
 3. Clear the Base URL in Nexus as shown in Figure 6.8 under Application Server Settings.
-

At this point, edit the HTTPd configuration file for the www.somecompany.com virtual host. Include the following to expose Nexus via mod_proxy at <http://www.somecompany.com/>

```
ProxyRequests Off
ProxyPreserveHost On

<VirtualHost *:80>
    ServerName www.somecompany.com
    ServerAdmin admin@somecompany.com
    ProxyPass / http://localhost:8081/
    ProxyPassReverse / http://localhost:8081/
    ErrorLog logs/somecompany/nexus/error.log
    CustomLog logs/somecompany/nexus/access.log common
</VirtualHost>
```

If you just wanted to continue to serve Nexus at the /nexus context path, you would not change the nexus-webapp-context-path in and you would include the context path in your ProxyPass and ProxyPassReverse

```
ProxyPass /nexus/ http://localhost:8081/nexus/
ProxyPassReverse /nexus/ http://localhost:8081/nexus/
```

Apache configuration is going to vary based on your own application's requirements and the way you intend to expose Nexus to the outside world. If you need more details about Apache httpd and mod_proxy, please see <http://httpd.apache.org>

3.9 Installing the Nexus WAR

The Nexus Open Source WAR has been successfully tested in Tomcat, Jetty, and Resin. If you need to run Nexus under Glassfish, please see Section 3.9.1. To download the Nexus Open Source WAR, go to <http://nexus.sonatype.org/using/download.html>. Click on the Download Site link and then download the Nexus WAR. Once you have downloaded the Nexus Open Source WAR, you can install it in a servlet container or application server.

The process for installing a WAR in an servlet container or application server is going to vary for each specific application. Often, this installation process is as simple as dropping a WAR file in a special directory and restarting the container. For example, to install the Nexus WAR in Tomcat, drop the nexus-2.0.6.war file in `$TOMCAT_HOME/webapps` and restart your Tomcat instance. Assuming that Tomcat is configured on port 8080 once Tomcat is started, Nexus will be available on <http://localhost:8080/nexus-2.0.6>

If you would like a less verbose URL, copy `nexus-2.0.6.war` to a file named `nexus.war` before copying the distribution to `$TOMCAT_HOME/webapps`

Note

When installing Nexus as a WAR in an application server or servlet container, it automatically creates a `sonatype-work` directory in the home directory of the user running the application server. This directory contains all of the necessary configuration and repository storage for Nexus.

3.9.1 Running the Nexus WAR in Glassfish

There is a known logging configuration conflict between the Nexus Open Source WAR and Glassfish. To address this conflict, you will need to modify the `log4j.properties` file and change a few of the directives that print messages to the console. The `log4j.properties` file is located in `~/sonatype-work/conf/log4j.properties`. This file is created the first time you attempt to deploy Nexus to Glassfish. To create this file, deploy Nexus to Glassfish. While this initial deployment will fail due to a logging conflict, the act of deploying Nexus to Glassfish will create the working directory in `~/sonatype-work` and the `log4j.properties`

Edit this `log4j.properties` file as follows:

1. On the first line, change "`log4j.rootLogger=INFO, console,`" to "`log4j.rootLogger=INFO, logfile`"
2. Remove the last three lines starting with "`log4j.appenders.console`"

These changes will instruct Log4J not to print messages to the console. Once you make these edits, restart Glassfish.

3.10 Installing a Nexus Professional License

To install a Nexus Professional license, click on Licensing in the Enterprise menu. Clicking on Licensing will bring up the panel shown in Figure 3.5. To upload your Nexus Professional license, click on `Browse...`, select the file, and click on `Upload`.



Figure 3.5: Nexus Professional Licensing Panel

Once you have selected a license and uploaded it to Nexus, Nexus Professional will display a dialog box with the Nexus Professional End-user License Agreement as shown in Figure 3.6. If you agree with the terms and conditions, click on "I Agree".

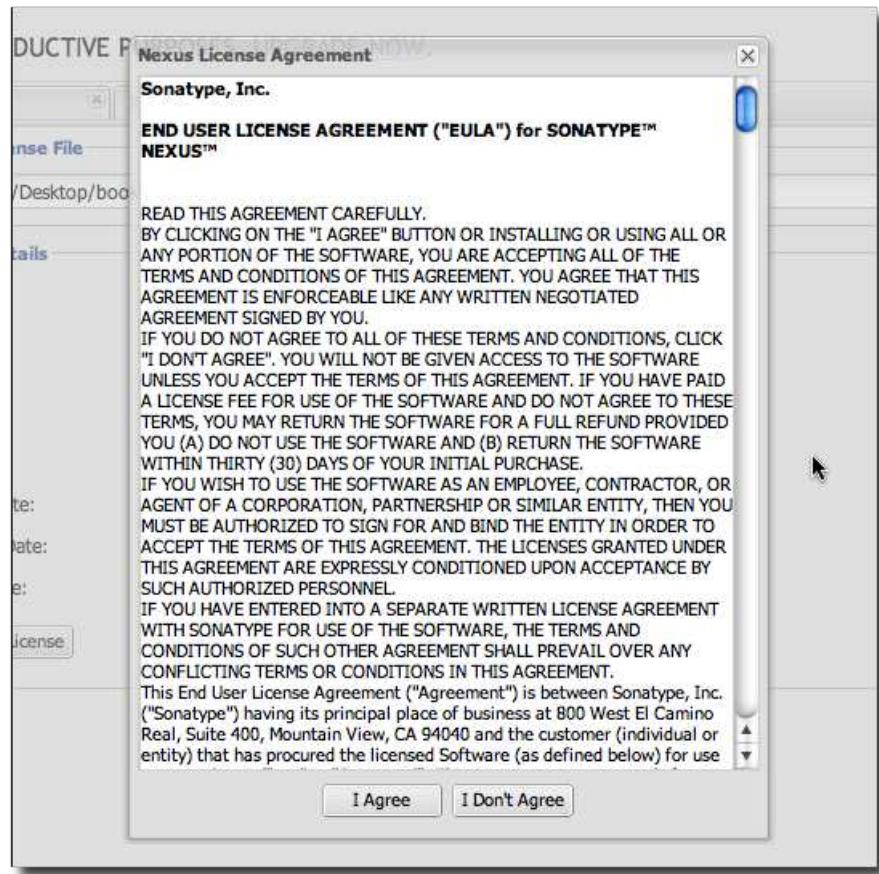


Figure 3.6: Nexus Professional End-user License Agreement

Once you have agreed to the terms and conditions contained in the End User License Agreement, Nexus Professional will then display a dialog box confirming the installation of a Nexus Professional license as shown in Figure 3.7.

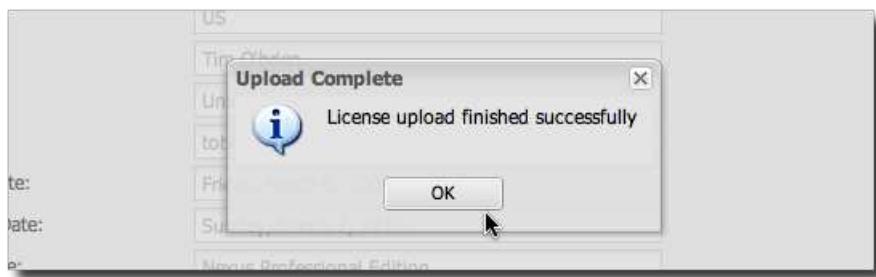


Figure 3.7: License Upload Finished Dialog

If you need to move your Nexus Professional license, you can click on the "Uninstall License" button at the bottom of the Licensing Panel. Clicking on this button will show the dialog in Figure 3.8 which confirms that you want to uninstall a license.

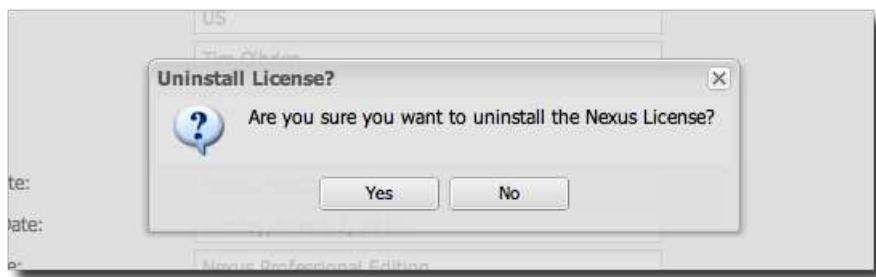


Figure 3.8: Uninstall License Confirmation Dialog

Clicking Yes in this dialog box will uninstall the license from Nexus Professional and display another dialog which confirms that the license has been successfully uninstalled.

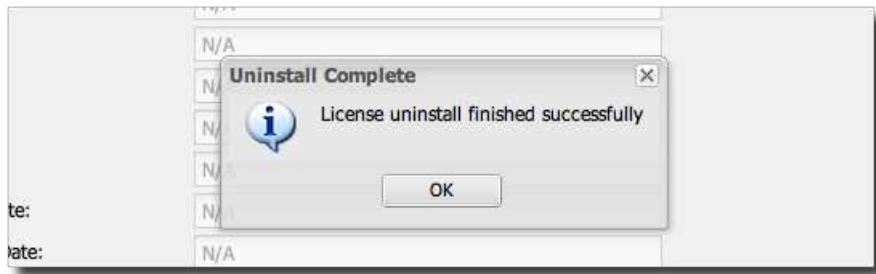


Figure 3.9: License Uninstall Completed Dialog

3.10.1 Evaluation Expiration

If you have downloaded the Nexus Professional evaluation, you will have 14 days to evaluate the product. After the expiration of this 14 day evaluation period, Nexus Professional will revert back to Nexus Open Source. To reactivate Nexus Professional after the end of your evaluation period, contact a Sonatype sales representative at sales@sonatype.com

3.11 Nexus Directories

The following sections describe the various directories that are a part of any Nexus installation. When you install Nexus Open Source or Nexus Professional, you are creating two directories: a directory which contains the Nexus runtime and application often symlinked as `nexus` and a directory which contains your own configuration and data - `sonatype-work/nexus`. When you upgrade to a newer version of Nexus, you replace the Nexus application directory and retain all of your own custom configuration and repository data in `sonatype-work/`

3.11.1 Sonatype Work Directory

The Sonatype Work directory `sonatype-work` is installed as a sibling to the `nexus` application directory, and the location of this directory can be configured via the `nexus.properties` file which is described in Section 3.11.2. Figure 3.10 shows the Sonatype Nexus work directory with some of its sub-directories.

```
$ tree sonatype-work/ -L 2
sonatype-work/
└── nexus
    ├── access
    ├── aether-local-repository
    ├── broker
    ├── conf
    ├── error-report-bundles
    ├── health-check
    ├── indexer
    ├── logs
    ├── nuget
    ├── plugin-repository
    ├── proxy
    ├── storage
    ├── template-store
    ├── timeline
    └── trash
    README.txt
```

Figure 3.10: The Sonatype Work Directory

The Sonatype Work Nexus directory `sonatype-work/nexus/` contains a number of sub-directories. Depending on the plugins installed and used some directories may or may be not present in your installation:

access/

This directory contains a log of all IP addresses accessing Nexus. The data can be viewed by clicking on Active Users Report in the Administration - Licensing tab in the Nexus user interface.

aether-local-repository/

This holds temporary files created when running Maven dependency queries in the user interface.

backup/

If you have configured a scheduled job to backup Nexus configuration, this directory is going to contain a number of ZIP archives that contain snapshots of Nexus configuration. Each ZIP file contains the contents of the `conf/` directory. (Automated backups are a feature of Nexus Professional.)

broker/

The broker directory and its sub-directories contains the storage backend for the Smart Proxy messaging component.

conf/

This directory contains the Nexus configuration. Settings that define the list of Nexus repositories, the logging configuration, the staging and procurement configuration, and the security settings are all captured in this directory.

conf/keystore/

Contains the automatically generated key used to identify this Nexus instance for Smart Proxy usage

error-report-bundles/

Contains the bundled archives of data assembled for problem reporting.

health-check/

Holds cached reports from the Repository Health Check plugin.

indexer/

Contains a Nexus index for all repositories and repository groups managed by Nexus. A Nexus index is a Lucene index which is the standard for indexing and searching a Maven repository. Nexus maintains a local index for all repositories, and can also download a Nexus index from remote repositories.

logs/

The nexus.log file that contains information about a running instance of Nexus. This directory also contains archived copies of Nexus log files. Nexus log files are rotated every day. To reclaim disk space, you can delete old log files from the logs/

nuget/

Contains the database supporting queries against NuGet repositories used for .NET package support in Nexus.

p2/

If you are using the P2 repository management features of Nexus Professional, this directory contains a local cache of P2 repository artifacts.

plugin-repository/

This directory contains any additionally installed plugins from third parties as documented in Section [3.12](#).

proxy/

Stores data about the files contained in a remote repository. Each proxy repository has a sub-directory in the proxy/attributes/ directory and every file that Nexus has interacted with in the remote repository has an XML file which captures such data as the: last requested timestamp, the remote URL for a particular file, the length of the file, and the digests for a particular file among other things. If you need to backup the local cached contents of a proxy repository, you should also back up the contents of the proxy repository's directory under proxy/attributes/

storage/

Stores artifacts and metadata for Nexus repositories. Each repository is a sub-directory which contains the artifacts in a repository. If the repository is a proxy repository, the storage directory will contain locally cached artifacts from the remote repository. If the repository is a hosted repository, the storage directory will contain all artifacts in the repository. If you need to backup the contents of a repository, you should backup the contents of the storage directory.

template-store/

Contains templates for default repositories. If you examine the XML files in this directory, you will see that they contain default templates for each different type of repository. For example, the repository-default_proxy_release.xml file contains defaults for a Proxy repository with a release policy.

timeline/

Contains an index which Nexus uses to store events and other information to support internal operations. Nexus uses this index to store feeds and history.

trash/

If you have configured scheduled jobs to remove snapshot artifacts or to delete other information from repositories, the deleted data will be stored in this directory. To empty this trash folder, view a list of Nexus repositories, and then click on the Trash icon in the Nexus user interface.

The conf/ directory contains a number of files which allow for configuration and customization of Nexus. All of the files contained in this directory are altered by the Nexus administrative user interface. While you can change the configuration settings contained in these files with a text editor, Sonatype recommends that you modify the contents of these files using the Nexus administrative user interface. Depending on your Nexus version and the installed plugins the complete list of files may differ slightly.

broker.groovy

A groovy script for configuring low level properties for Smart Proxy.

capabilities.xml

Further Smart Proxy backend configuration.

healthcheck.properties

Configuration for the Repository Health Check.

log4j.properties, logback.properties, logback.xml and logback-* .xml

Contains logging configuration. If you need to customize the detail of log messages, the frequency of log file rotation, or if you want to connect your own, custom Log4J appenders, you would alter this configuration file.

lvo-plugin.xml

Contains configuration for the latest version plugin. This XML file contains the location of the properties file which Nexus queries to check for a newer version of Nexus.

nexus.xml

The bulk of the configuration of Nexus is contained in this file. This file maintains a list of repositories, and all server-wide configuration like the SMTP settings, security realms, repository groups, targets, and path mappings.

pgp.xml

Contains PGP key server configuration.

nexus-obr-plugin.properties

Contains configuration for the Nexus OSGi Bundle repository plugin in Nexus Professional.

procurement.xml

Contains configuration for the Nexus Procurement plugin in Nexus Professional.

security-configuration.xml

Contains global security configuration.

security.xml

Contains security configuration about users and roles.

staging.xml

Contains configuration for the Nexus Staging Plugin in Nexus Professional.

3.11.2 Nexus Configuration Directory

After installing Nexus and creating the nexus symlink as described earlier, your nexus folder contains another conf directory. This directory contains configuration for the Jetty servlet container. You will only need to modify the files in this directory if you are customizing the configuration of Jetty servlet container, or the behaviour of the scripts that start Nexus.

The files and folders contained in this directory are:

classworlds.conf

Defines the order in which resources and classes are loaded from the classpath. It is unlikely that even the most advanced Nexus users will ever need to customize the contents of this file.

nexus.properties

This file contains configuration variables which control the behaviour of Nexus and the Jetty servlet container. If you are customizing the port and host that Nexus will listen to, you would change the application-port and application-host properties defined in this file. If you wanted to customize the location of the Sonatype work directory, you would modify the value of the nexus-work property in this configuration file. Changing nexus-webapp-context-path allows you to configure the server context path Nexus will be available at.

jetty.xml

If this file is present in the conf/ directory, it will be used to configure Jetty.

The conf/examples/ directory contains sample Jetty configuration files which can be used to customize the behaviour of the Jetty servlet container:

jetty.xml

contains a jetty.xml sample with no customizations. This sample file listens on the "application-port" defined in nexus.properties

jetty-ajp.xml

Contains a jetty.xml sample which will configure Nexus to listen on an AJP port 8009. This configuration can be used if you are proxying your Nexus server with web server which understands the AJP protocol such as Apache httpd with the mod_proxy_ajp module.

jetty-dual-ports-with-ssl.xml

Contains a jetty.xml sample which configures Nexus to listen on both the "application-port" and "application-port-ssl" (as defined in nexus.properties). This sample configuration also contains the SSL redirect rule.

jetty-faster-windows.xml

Contains a jetty.xml sample which configures a response buffer size that will address performance issues on Windows 2003 Server, for more information about this fix see [the Jetty Wiki](#)

jetty-header-buffer.xml

Contains a jetty.xml sample which increases the headerBufferSize to 8k from the default of 4k. Documentation about the header buffer size can be found on [the Jetty Wiki](#)

jetty-simple-https-proxy.xml

Contains a jetty.xml sample which should be used if you are proxying a Nexus instance with a web server that is handling SSL. For example, if you were proxying Nexus with Apache httpd server using mod_ssl you would use this configuration to configure the Jetty RewriteHandler

jetty-ssl.xml

Contains a jetty.xml sample which will only serve SSL encrypted content from "application-port" (as defined in nexus.properties)

The conf/examples/proxy-https/ directory contains two files: apache2.conf and jetty.xml contains sample mod_proxy directives to configure Apache httpd to handle SSL.

3.12 Installing Additional Plugins

Nexus Open Source and Nexus Professional are built using a plugin architecture, where Professional simply includes additional plugins. Besides the default installed plugins you can install optional plugins shipped with Nexus as well as plugins you or somebody else created.

Optional plugins supplied by Sonatype can be found in the directory \$NEXUS_HOME/nexus/WEB-INF/optional-plugins. To install any of these simply copy the folder containing the desired plugin into \$NEXUS_HOME/nexus/WEB-INF/plugin-repository

Plugins supplied by third parties like the [Nexus Yum Plugin](#) and others, are installed by copying the folder with the plugin code into sonatype-work/nexus/plugin-repository.

After a restart of Nexus the new plugins will be active and ready to use. Upgrades are done by simply copying over the newer plugin and removing the older one.

Chapter 4

Configuring Maven to Use Nexus

4.1 Introduction

To use Nexus, you will configure Maven to check Nexus instead of the public repositories. To do this, you'll need to edit your mirror settings in your `~/.m2/settings.xml` file. First, we're going to demonstrate how to configure Maven to consult your Nexus installation instead of retrieving artifacts directly from the Maven Central repository. After we override the central repository and demonstrate that Nexus is working, we'll circle back to provide a more sensible set of settings that will cover both releases and snapshots.

4.2 Configuring Maven to Use a Single Nexus Group

If you are adopting Nexus for internal development you should configure a single Nexus group which contains both releases and snapshots. To do this, add snapshot repositories to your public group, and add the following mirror configuration to your Maven settings in `~/.m2/settings.xml`

Configuring Maven to Use a Single Nexus Group

```
<settings>
  <mirrors>
    <mirror>
```

```
<!--This sends everything else to /public -->
<id>nexus</id>
<mirrorOf>*</mirrorOf>
<url>http://localhost:8081/nexus/content/groups/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 Nexus Group](#) we have defined a single profile: nexus profile is configured to download from the central repository 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 Nexus group. The nexus group is then listed as an active profile in the activeProfiles element.

4.3 Adding Custom Repositories for Missing Dependencies

If you've configured your Maven settings.xml to list the Nexus public group as a mirror for all repositories, you might encounter projects which are unable to retrieve artifacts from your local Nexus installation. This usually happens because you are trying to build a project which has defined a custom set of repositories and snapshotRepositories in a pom.xml. When you encounter a project which contains a custom repository element in a pom.xml add this repository to Nexus as a new proxy repository and then add the new proxy repository to the public group.

4.4 Adding a New Repository

To add a repository, log into Nexus as an Administrator, and click on the Repositories link in the left-hand navigation menu in the Views/Repositories section as displayed in Figure 4.1.

Clicking on this link should bring up a window that lists all of the repositories which Nexus knows about. You'll then want to create a new proxy repository. To do this, click on the Add link that is directly above the list of repositories. When you click the Add button, click the down arrow directly to the right of the word Add, this will show a drop-down which has the options: Hosted Repository, Proxy Repository, Virtual Repository, and Repository Group. Since you are creating a proxy repository, click on Proxy Repository.

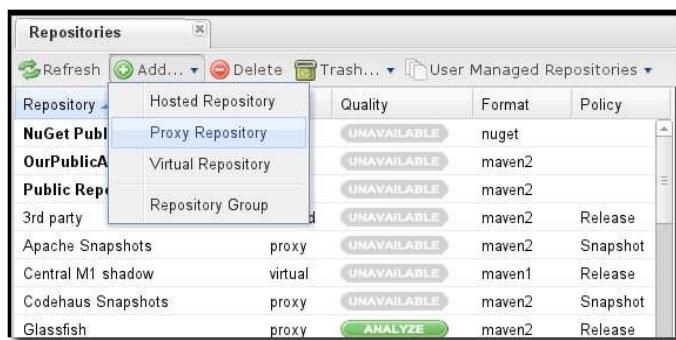


Figure 4.1: Creating a New Proxy Repository

Once you do this, you will see a screen resembling Figure 4.2. Populate the required fields Repository ID and the Repository Name. The Repository ID will be part of the URL used to access the repository, so it

is recommended to avoid characters that could cause problems. Set the Repository Policy to "Release", and the Remote Storage Location to the public URL of the repository you want to proxy.

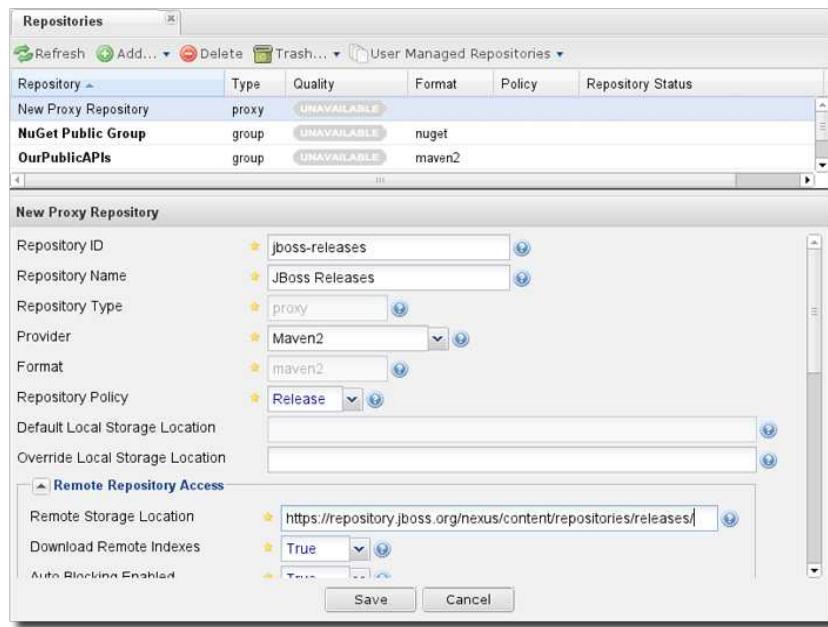


Figure 4.2: Configuring a Proxy Repository

Once you've filled out this screen, click on the Save button. Nexus will then be configured to proxy the repository.

4.5 Adding a Repository to a Group

Next you will need to add the new repository to the Public Repositories Nexus Group. To do this, click on the Repositories link in the left-hand navigation menu in the Views/Repositories section. Nexus lists Groups and Repositories in the same list so click on the public group. After clicking on the Public Repositories group, you should see the Browse and Configuration tabs in the lower half of the Nexus window.

Note

If you click on a repository or a group in the Repositories list and you do not see the Configuration tab, this is because your Nexus user does not have administrative privileges. To perform the configuration tasks outlined in this chapter, you will need to be logged in as a user with administrative privileges.

Clicking on the Configuration tab will bring up a screen which looks like Figure 4.3.

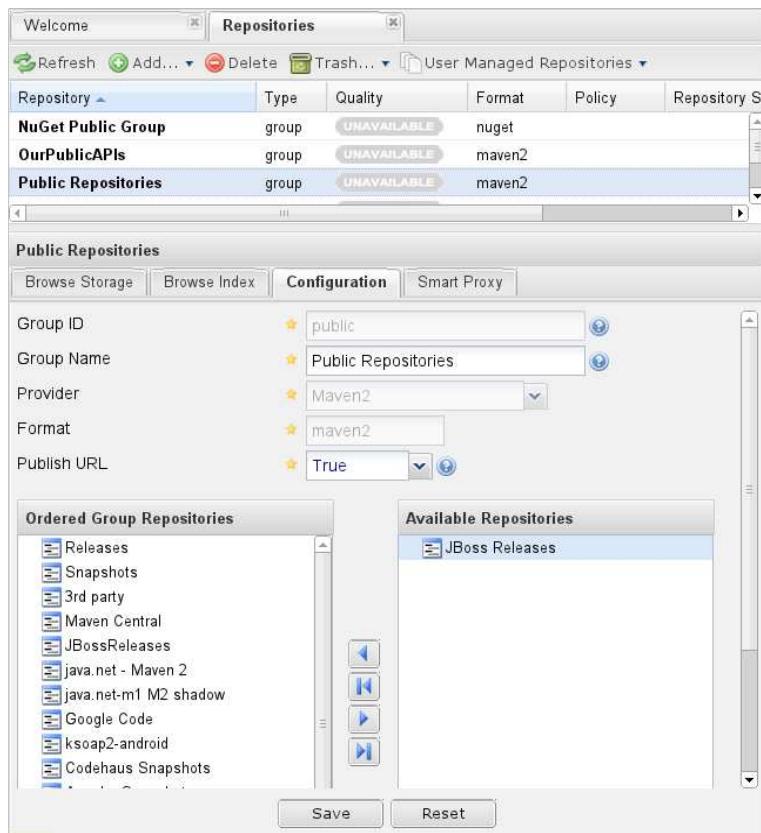


Figure 4.3: Adding New Repositories to a Nexus Group

To add the new repository to the public group, find the repository in the Available Repositories list on the right, click on the repository you want to add and drag it to the left to the Ordered Group Repositories list. Once the repository is in the Ordered Group Repositories list you can click and drag the repository within that list to alter the order in which a repository will be searched for a matching artifact.

Note

Nexus makes use of an interesting Javascript widget library named [ExtJS](#). ExtJS provides for a number of interesting UI widgets that allow for rich interaction like the drag-drop UI for adding repositories to a group and reordering the contents of a group.

In the last few sections, you learned how to add a new custom repositories to a build in order to download artifacts which 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 Nexus repository manager to add the two repositories to the public group. If all of the developers are configured to point to the public group in Nexus, 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.

Chapter 5

Using Nexus

5.1 Introduction

Nexus provides anonymous access for users who only need to search repositories, browse repositories, and peruse the system feeds. This anonymous access level changes the navigation menu and some of the options available when you right-click on a repository. This read-only access displays the user interface shown in Figure 5.1.

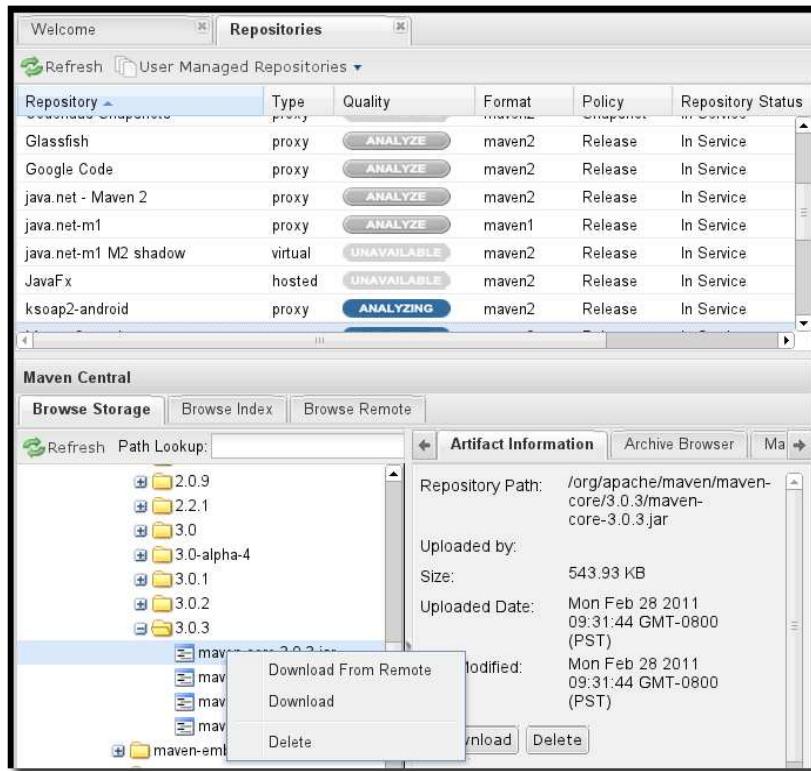


Figure 5.1: Nexus Interface for Anonymous Users

5.2 Browsing Repositories

One of the most straightforward uses of the Nexus is to browse the structure of a repository. If you click on the Repositories menu item in the Views/Repositories menu, you should see the following display. The top-half of Figure 5.2 shows you a list of groups and repositories along with the type of the repository and the repository status. To browse the artifacts that are stored in a local Nexus instance, click on the Browse Storage tab for a repository as shown in Figure 5.2.

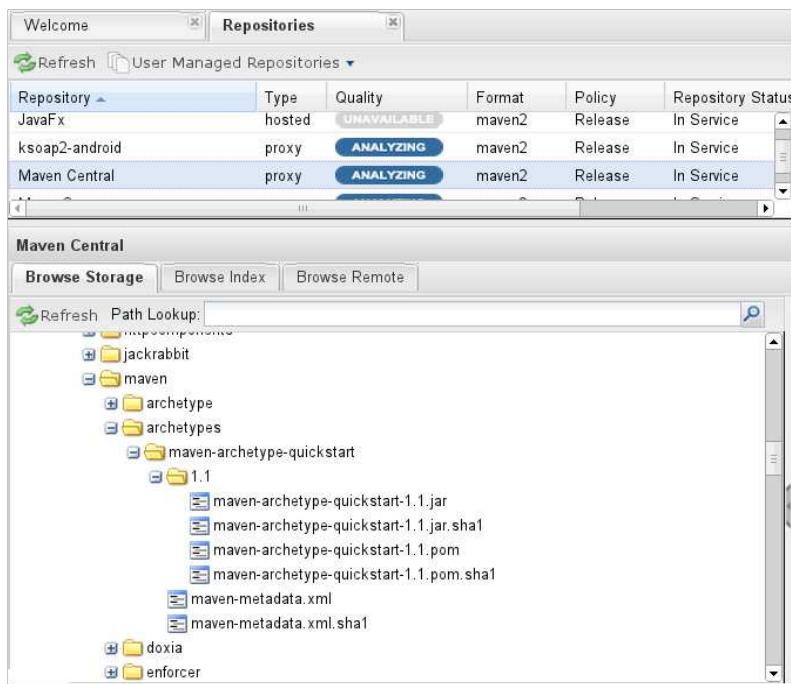


Figure 5.2: Browsing a Repository Storage

When you are browsing a repository, you can right click on any file and download it directly to your browser. This allows you to retrieve specific artifacts manually, or examine a POM file in the browser.

Note

When browsing a remote repository you might notice that the tree doesn't contain all of the artifacts in a repository. When you browse a proxy repository, Nexus is displaying the artifacts which have been cached locally from the remote repository. If you don't see an artifact you expected to see through Nexus, it only means that Nexus has yet to cache the artifact locally. If you have enabled remote repository index downloads, Nexus will return search results that may include artifacts not yet downloaded from the remote repository. Figure 5.2, is just an example, and you may or may not have the example artifact available in your installation of Nexus.

A Nexus proxy repository acts as a local cache for a remote repository, in addition to downloading and caching artifacts locally, Nexus will also download an index of all the artifacts stored in a particular

repository. When searching or browsing for artifacts, it is often more useful to search and browse the repository index. To view the repository index, click on the Browse Index tab for a particular repository to load the interface shown in Figure 5.3.

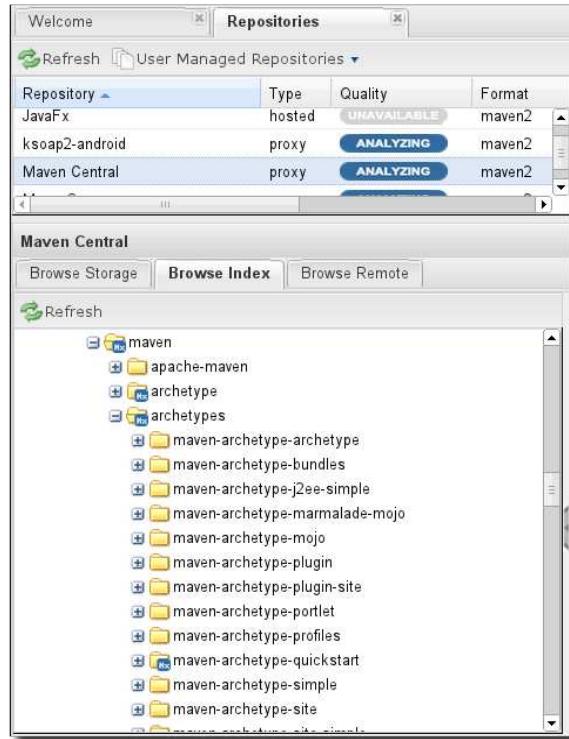


Figure 5.3: Browsing a Repository Index

As shown in Figure 5.3, if an artifact has been downloaded from a remote repository and cached in Nexus, the artifact or folder will display a small Nexus logo.

5.2.1 Viewing the Artifact Information

Once you located an archive in the repository index or storage the right hand panel will at minimum show the Artifact Information tab as visible in Figure 5.4. Besides showing details like the Repository Path, Size, Checksums, location of the artifact and other details you are able to download and delete the artifact with the respective buttons.



Figure 5.4: Viewing the Artifact Information

5.2.2 Viewing the Artifact Maven Information

If the artifact you are looking at in the browser is a Maven related artifact like a pom file or a jar you will see the Maven Information tab in the right hand panels. As visible in Figure 5.5 the GAV parameters are displayed above an XML snippet identifying the artifact that you can just cut and paste into a Maven pom.xml file.



Figure 5.5: Viewing the Maven Information

5.2.3 Using the Artifact Archive Browser

For binary artifacts like jar files Nexus displays an Archive Browser panel as visible in Figure 5.6 that allows you to view the contents of the archive. Clicking on individual files in the browser will download them.



Figure 5.6: Using the Archive Browser

5.2.4 Viewing the Artifact Dependencies

Nexus Professional provides you with the ability to browse an artifact's dependencies. Using the artifact metadata found in an artifact's POM, Nexus will scan a repository or a repository group and attempt to resolve and display an artifact's dependencies. To view an artifact's dependencies, browse the repository storage or the repository index, select an artifact (or an artifact's POM), and then click on the Maven Dependency tab.

On the Maven Dependency tab, you will see the following form elements:

Repository

When resolving an artifact's dependencies, Nexus will query an existing repository or repository group. In many cases it will make sense to select the same repository group you are referencing in your Maven Settings. If you encounter any problems during the dependency resolution, you need to make sure that you are referencing a repository or a group which contains these dependencies.

Mode

An artifact's dependencies can be listed as either a tree or a list. When dependencies are displayed in

a tree, you can inspect direct dependencies and transitive dependencies. This can come in handy if you are assessing an artifact based on the dependencies it is going to pull into your project's build. When you list dependencies as a list, Nexus is going to perform the same process used by Maven to collapse a tree of dependencies into a list of dependencies using rules to merge and override dependency versions if there are any overlaps or conflicts.

Once you have selected a repository to resolve against and a mode to display an artifact's dependencies, click on the Resolve button as shown in Figure 5.7. Clicking on this button will start the process of resolving dependencies, depending on the number of artifacts already cached by Nexus, this process can take anywhere from a few seconds to minute. Once the resolution process is finished, you should see the artifact's dependencies as shown in Figure 5.7.

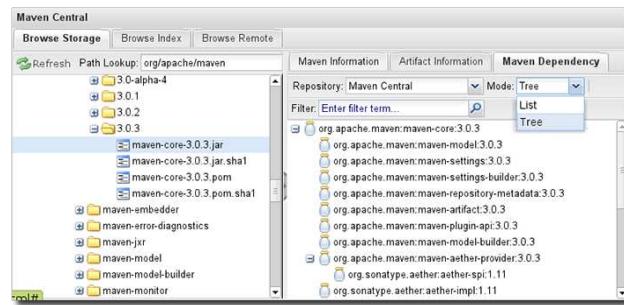


Figure 5.7: View an Artifact's Dependencies

Once you have resolved an artifact's dependencies, you can use the Filter text input to search for particular artifact dependencies. If you double click on a row in the tree or list of dependencies you can navigate to other artifacts within the Nexus interface.

5.3 Browsing Groups

Nexus contains ordered groups of repositories which allow you to expose a series of repositories through a single URL. More often than not, an organization is going to point Maven at the two default Nexus groups: Public Repositories and Public Snapshot Repositories. Most end-users of Nexus are not going to know what artifacts are being served from what specific repository, and they are going to want to be able to browse the Public Repository. To support this use case, Maven allows you to browse the contents of a Nexus Group as if it were a single merged repository with a tree structure. Figure 5.8, shows the browsing storage interface for a Nexus Group. There is no difference to the user experience of browsing a Nexus

Group vs. browsing a Nexus Repository.

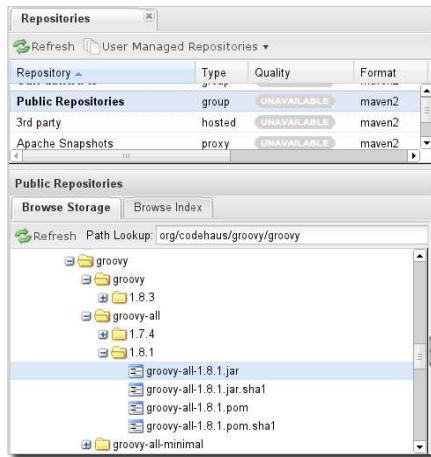


Figure 5.8: Browsing a Nexus Group

When browsing a Nexus group's storage, you are browsing the underlying storage for all of the repositories which are contained in a group. If a Nexus group contains proxy repositories, the Browse Storage tab will show all of the artifacts in the Nexus group that have been download from the remote repositories. To browse and search all artifacts available in a Nexus group, click on the Browse Index tab to load the interface shown in Figure 5.9.

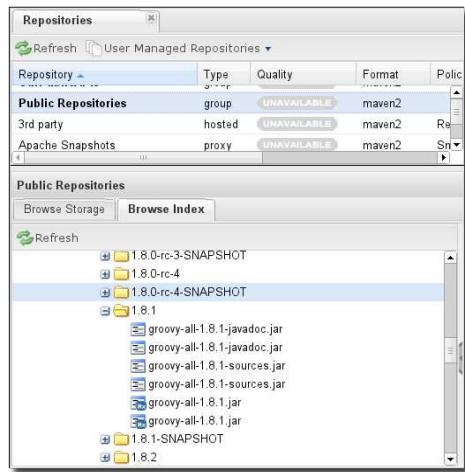


Figure 5.9: Browsing a Nexus Group Index

5.4 Searching for Artifacts

5.4.1 Search Overview

In the left-hand navigation area, there is an Artifact Search text field next to a magnifying glass. To search for an artifact by groupId or artifactId, type in some text and click the magnifying glass. Typing in the search term "junit" and clicking the magnifying glass should yield a search result similar to Figure 5.10.

The screenshot shows the Nexus search interface. At the top, there's a search bar with 'junit' typed in. Below it is a table with columns: Group, Artifact, Version, Mos..., and Download. The table lists several artifacts, including junit, junitperf, junit-addons, junit, and com.googlecode... The 'junit' row is highlighted. To the right of the table is a detailed view pane. The left side of the detailed view shows a tree structure of Maven Central repositories, with 'junit' expanded to show versions 3.7, 3.8, 3.8.1, 3.8.2, 4.0, 4.1, and 4.10. The '4.10' node is also expanded to show 'junit-4.10-javadoc', 'junit-4.10-sources', and 'junit-4.10.jar'. The right side of the detailed view pane shows the Maven Information for the selected 'junit' artifact: Group: junit, Artifact: junit, Version: 4.10, Extension: jar. Below this is an XML snippet of a dependency declaration:

```
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.10</version>
</dependency>
```

Figure 5.10: Results of an Artifact Search for "junit"

The groupId in the Group column and the artifactId in the Artifact column identify each row in the search results table. Each row represents an aggregation of all artifacts in this Group and Artifact coordinate.

The Version column displays the lastest version number available as well as a links to Show All Versions.

The Most Popluar Version column displays the version that has the most downloads by all users accessing the Central Repository. This data can help with the selection of an appropriate version to use for a particular artifact.

The Download column displays direct links to all the artifacts available for the latest version of this artifacts. A typical list of downloadable artifacts would include the Java archive (jar), the Maven pom.xml file (pom), a Javadoc archive (javadoc.jar) and a Sourcecode archive (sources.jar), but other download options are also added if more artifacts are available. Click on the link to download an artifact.

Each of the columns in the search results table can be used to sort the table in Ascending or Descending order. In addition you can choose to add and remove columns with the sort and column drop down options visible in Figure 5.11.

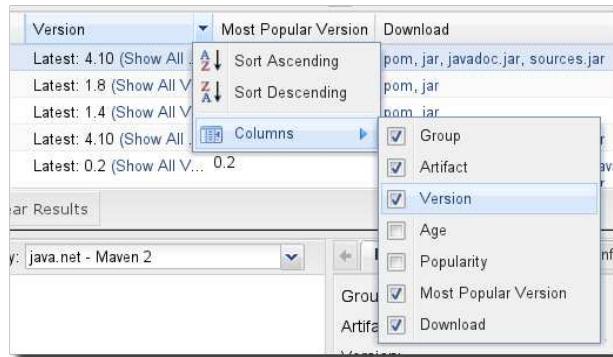


Figure 5.11: Sort and Column Options in the Search Results Table

The repository browser interface below the search results table will displays the artifact selected in the list in the repository structure with the same information panels available documented in Section 5.2. An artifact could be present in more than one repository. If this is the case, click on the value next to "Viewing Repository" to switch between multiple matching repositories.

Warning

 Let me guess? You installed Nexus, ran to the search box, typed in the name of a group or an artifact, pressed search, and saw absolutely nothing. No results. Nexus isn't going to retrieve the remote repository indexes by default, you need to activate downloading of remote indexes for the three proxy repositories that Nexus ships with. Without these indexes, Nexus has nothing to search. Find instructions for activating index downloads in Section 6.2.

5.4.2 Advanced Search

Clicking on the (Show All Versions) link in the Version column visible in Figure 5.10 will kick off an Advanced Search by the groupId and artifactId of the row and result in a view similar to Figure 5.12.

The screenshot shows a search interface with the following details:

Group	Artifact	Version	Age	Popularity	Download
junit	junit	4.8.2	1.4 yrs	<div style="width: 80%;">80%</div>	pom, jar, sources.jar, javadoc.jar
junit	junit	3.8.2	4.8 yrs	<div style="width: 60%;">60%</div>	pom, jar, sources.jar, javadoc.jar
junit	junit	4.10	141 d	<div style="width: 50%;">50%</div>	pom, sources.jar, javadoc.jar, jar
junit	junit	3.8.1	4.8 yrs	<div style="width: 40%;">40%</div>	pom, jar, sources.jar
junit	junit	4.8.1	2.0 yrs	<div style="width: 30%;">30%</div>	pom, jar, sources.jar
junit	junit	4.4	4.5 yrs	<div style="width: 20%;">20%</div>	pom, jar, sources.jar, javadoc.jar

Displaying Top 18 records | Clear Results

Figure 5.12: Advanced Search Results for a GAV Search Activated by the Show All Versions Link

The header for the Advanced Search contains a selector for the type of search and one or more text input fields to define a search and a button to run a new search with the specified parameters.

The search results table contains one row per Group (groupId), Artifact (artifactId) and Version(version).

In addition the Age column displays the age of the artifacts being available on the Central Repository. Since most artifacts are published to the Central Repository when released, this age gives you a good indication of the actual time since the release of the artifact.

The Popularity column shows a relative popularity as compared to the other results in the search table. This can give you a good indication on the take up of a new release. For example if a newer version has a high Age value, but a low Popularity compared to an older version, you might want to check the upstream project and see if there is any issues stopping other users from upgrading that might affect you as well. Another reason could be that the new version does not provide significant improvements to warrant an upgrade for most users.

The Download column provide download links for all the available artifacts.

The following advanced searches are available:

Keyword Search

Identical to the Artifact Search in the left hand navigation, this search will look for the specified strings in the groupId and artifactId.

Classname Search

Rather than looking at the coordinates of an artifact in the repository, the Classname Search will look at the contents of the artifacts and look for Java classes with the specified name. For example try a search for a classname of "Pair" to see how many library authors saw a need to implement such a class, saving you from potentially implementing yet another version.

GAV Search

The GAV search allows a search using the Maven coordinates of an artifact. These are Group (groupId), Artifact (artifactId), Version (version), Packaging (packaging) and Classifier (classifier). At a minimum you need to specify a Group, Artifact or Version in your search. An example search would be with an Artifact "guice" and a Classifier "no_aop".

Checksum Search

Sometimes it is necessary to determine the version of a jar artifact in order to migrate to a qualified version. When attempting this and neither the filename nor the contents of the manifest file in the jar contain any useful information about the exact version of the jar you can use Checksum Search to identify the artifact. Create a sha1 checksum, e.g. with the `shasum` command available on Linux, and use the created string in a Checksum search. This will return one result, which will provide you with the GAV coordinates to replace the jar file with a dependency declaration.

Tip

The Checksum Search can be a huge time saver when migrating a legacy build system, where the used libraries are checked into the version control system as binary artifacts with no version information available.

5.4.3 Nexus OpenSearch Integration

OpenSearch is a standard which facilitates searching directly from your browser's search box. If you are using Internet Explorer 7+ or Firefox 2+ you can add any Nexus instance as an OpenSearch provider. Then you can just type in a search term into your browser's search field and quickly search for Maven artifacts. To configure OpenSearch, load Nexus in a browser and then click on the drop-down next to the search tool that is embedded in your browser. Figure 5.13 shows the Add Nexus option that is present in Firefox's OpenSearch provider drop-down.

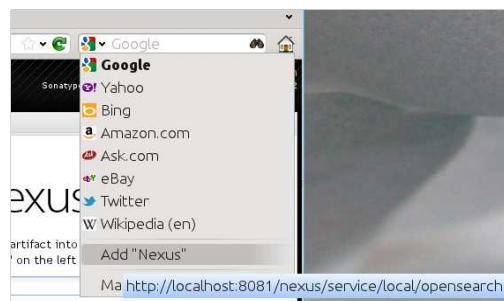


Figure 5.13: Configuring Nexus as an OpenSearch Provider

Once you have added Nexus to the list of OpenSearch providers, click on the drop-down next to the search term and select Nexus (localhost) from the list of OpenSearch providers. Type in a groupId, artifactId, or portion of a Maven identifier and press enter. Your opensearch-friendly web browser will then take you to the search results page of Nexus displaying all the artifacts that match your search term.

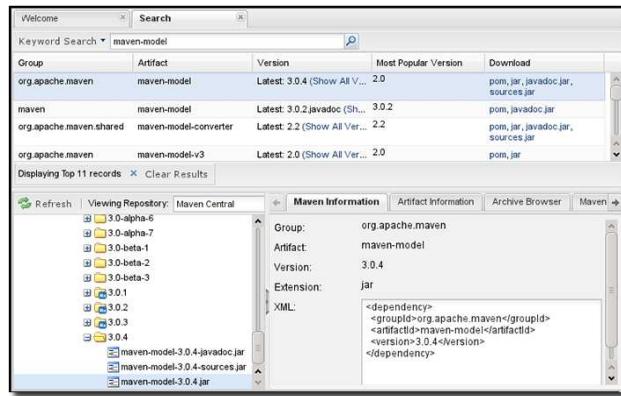


Figure 5.14: OpenSearch Search Results in Nexus

Once, you have configured your browser to use Nexus as an OpenSearch provider, searching for a Maven artifact is as simple as typing in a groupId or artifactId, selecting Nexus from the drop-down shown in Figure 5.15, and performing a search.



Figure 5.15: Nexus Available as an Option in the Firefox OpenSearch Provider List

5.5 Uploading Artifacts

When your build makes use of proprietary or custom dependencies which are not available from public repositories, you will often need to find a way to make them available to developers in a custom Maven repository. Nexus ships with a pre-configured 3rd Party repository that was designed to hold 3rd Party dependencies which are used in your builds. To upload artifacts to a repository, select a hosted repository in the Repositories panel and then click on the Artifact Upload tab. Clicking on the Artifact Upload tab will display the tab shown in Figure 5.16.

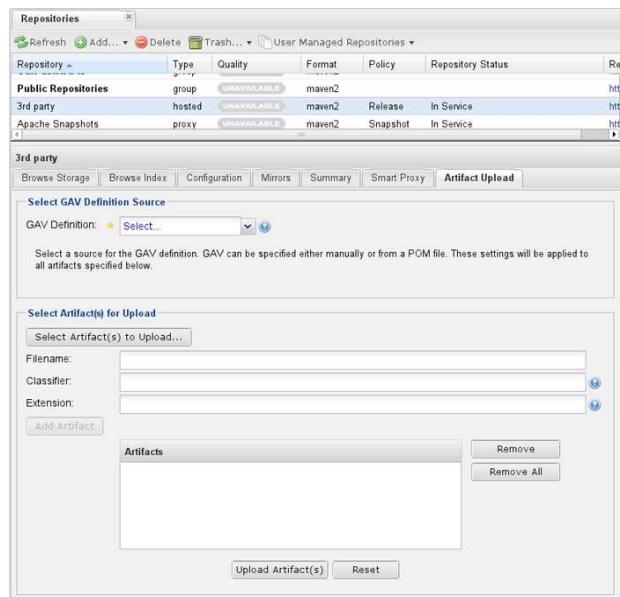


Figure 5.16: Artifact Upload Form

To upload an artifact, click on Select Artifact(s) for Upload... and select one or more artifacts from the file-system to upload. Once you have selected an artifact, you can modify the classifier and the extension before clicking on the Add Artifact button. Once you have clicked on the Add Artifact button, you can then configure the source of the Group, Artifact, Version (GAV) parameters.

If the artifact you are uploading is a JAR file that was created by Maven it will already have POM information embedded in it. If you are uploading a JAR from a vendor you will likely need to set the Group Identifier, Artifact Identifier, and Version manually. To do this, select GAV Parameters from the GAV Definition drop-down at the top of this form. Selecting GAV Parameters will expose a set of form fields

which will let you set the Group, Artifact, Version, and Packaging of the artifacts being uploaded.

If you would prefer to set the Group, Artifact, and Version from a POM file associated with the uploaded artifact, select From POM in the GAV Definition drop-down. Selecting From POM in this drop-down will expose a button labelled "Select POM to Upload". Once a POM file has been selected for upload, the name of the POM file will be displayed in the form field below this button.

The Artifact Upload panel supports multiple artifacts with the same Group, Artifact, and Version identifiers. For example, if you need to upload multiple artifacts with different classifiers, you may do so by clicking on Select Artifact(s) for Upload and Add Artifact multiple times.

5.6 Browsing System Feeds

Nexus provides feeds that capture system events, you can browse these feeds by clicking on System Feeds under the View menu. Clicking on System Feeds will show the panel in Figure 5.17. You can use these simple interface to browse the most recent reports of artifact deployments, cached artifacts, broken artifacts, and storage changes that have occurred in Nexus.

The screenshot shows the 'System Feeds' interface in Nexus. At the top, there are two buttons: 'Refresh' and 'Subscribe'. Below them is a table with columns for 'Feed' and 'URL'. The 'Feed' column lists various system events, and the 'URL' column provides the corresponding URLs for each event. The table has a scroll bar on the right side. Below the table, there is a section titled 'New artifacts in all Nexus repositories (cached or deployed)' with a table of its own. This table has columns for 'Title', 'Date', and 'Details'. It lists three artifact deployment entries, each with a timestamp and a detailed description of the deployment action. The entire interface is contained within a window with a title bar and a close button.

Feed	URL
Authentication and Authorization Events	http://localhost:8081/nexus/service/local...
Broken artifacts in all Nexus repositories (checksum error...)	http://localhost:8081/nexus/service/local...
Broken files in all Nexus repositories (checksum errors,...)	http://localhost:8081/nexus/service/local...
Error and Warning events	http://localhost:8081/nexus/service/local...
New artifacts in all Nexus repositories (cached or deployed)	http://localhost:8081/nexus/service/local...
New cached artifacts in all Nexus repositories (cached)	http://localhost:8081/nexus/service/local...

Title	Date
com.simpligility.maven:stagingexample:1.6.1 The artifact 'com.simpligility.maven:stagingexample:1.6.1' in repository 'Releases' was deployed.	1/25 1:00 pm
com.simpligility.maven:stagingexample:1.6 The artifact 'com.simpligility.maven:stagingexample:1.6' in repository 'Releases' was deployed. Action was initiated by user "admin". Request originated from IP address 127.0.0.1.	1/25 12:52 pm
com.simpligility.maven:stagingexample:1.6- 20120125.205105-2 The artifact 'com.simpligility.maven:stagingexample:1.6-20120125.205105-2' in repository 'Snapshots' was deployed. Action was initiated by user "admin". Request originated from IP address 127.0.0.1.	1/25 12:51 pm
com.simpligility.maven:stagingexample:1.6- 20120125.204958-1 The artifact 'com.simpligility.maven:stagingexample:1.6-20120125.204958-1' in repository 'Snapshots' was deployed. Action was initiated by user "admin". Request originated from IP address 127.0.0.1.	1/25 12:49 pm

Figure 5.17: Browsing Nexus System Feeds

These feeds can come in handy if you are working at a large organization with multiple development

teams deploying to the same instance of Nexus. In such an arrangement, all developers in an organization can subscribe to the RSS feeds for New Deployed Artifacts as a way to ensure that everyone is aware when a new release has been pushed to Nexus. Exposing these system events as RSS feeds also opens to the door to other, more creative uses of this information, such as connecting Nexus to external automated testing systems. To access the RSS feeds for a specific feed, select the feed in the System Feeds view panel and then click on the Subscribe button. Nexus will then load the RSS feed in your browser and you can subscribe to the feed in your favourite RSS

There are a number of system feeds available in the System Feeds view, and each has a URL which resembles the following URL

```
http://localhost:8081/nexus/service/local/feeds/recentlyChangedFiles
```

Where recentChanges would be replaced with the identifier of the feed you were attempting to read. Available system feeds include:

- Authentication and Authorization Events
- Broken artifacts in all Nexus repositories
- Broken files in all Nexus repositories
- Error and Warning events
- New artifacts in all Nexus repositories
- New cached artifacts in all Nexus repositories
- New cached files in all Nexus repositories
- New cached release artifacts in all Nexus repositories
- New deployed artifacts in all Nexus repositories
- New deployed files in all Nexus repositories
- New deployed release artifacts in all Nexus repositories
- New files in all Nexus repositories
- New release artifacts in all Nexus repositories
- Recent artifact storage changes in all Nexus repositories
- Recent file storage changes in all Nexus repositories
- Recent release artifact storage changes in all Nexus repositories
- Repository Status Changes in Nexus
- System changes in Nexus

5.7 System Files

The System Files is only visible to Administrative users under the Administrationmenu. Click on this option brings up the dialog shown in Figure 5.18. From this screen you can view the Nexus log file as well as the configuration files documented in Section 3.11.1.

The nexus.log file is the general application log for Nexus. Unless you are an administrative user, you might not have much interest in the information in this log. If you are trying to debug an error, or if you have uncovered a bug in Nexus, you can use this log viewer to diagnose problems with Nexus.

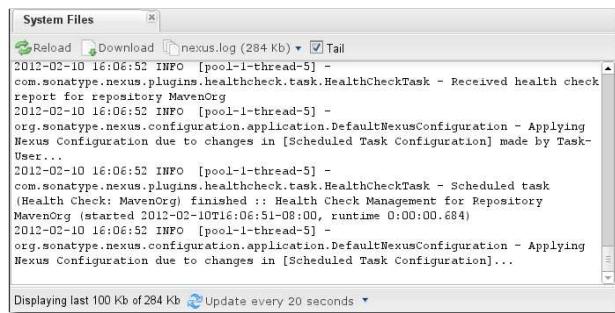


Figure 5.18: Browsing Nexus Logs and Configuration

You can load, view and download the various files by using the buttons and the dropdown to select the files to examine.

In Figure 5.18 there is a "tail" checkbox. If this box is checked, then Nexus will always show you the end of a log file. This can come in handy if you want to see a continuously updated log file. When this tail box is checked, a drop-down at the bottom of the panel allows you to set the update frequency. The contents of this drop-down are shown in Figure 5.19. If an update interval is selected, Nexus will periodically refresh the selected log file.

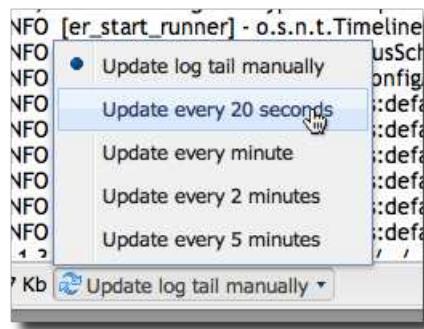


Figure 5.19: Selecting the Update Frequency when Tailing a Log File

5.8 Changing Your Password

If you have the appropriate security privilege, you will see an option to change your password in the left-hand side Security menu. To change your password, click on Change Password, supply your current password, and choose a new password. When you click on Change Password, your Nexus password will be changed.



Figure 5.20: Changing Your Nexus Password

5.9 Filing a Problem Report

If you encounter a problem with Nexus, you can use the Nexus UI to report a bug or file an issue against the Nexus project in Sonatype's JIRA instance.

To file a problem report, you will first need to sign up for an account on <http://issues.sonatype.org>. You can click on Report Problem in the left hand Helpmenu, supply your Sonatype JIRA credentials, and file a problem report. Supply your JIRA username and password along with a short title and a description as shown in the following figure.

When you file a Nexus problem report, Nexus will create a new issue in JIRA and attach your configuration and logs to the newly filed issue.



Figure 5.21: Generating a Nexus Problem Report

Chapter 6

Configuring Nexus

Many of the configuration screens shown in this section are only available to administrative users. Nexus allows the admin user to customize the list of repositories, create repository groups, customize server settings, and create routes or "rules" that Maven will use to include or exclude artifacts from a repository.

6.1 Customizing Server Configuration

In a production installation of Nexus, you'll probably want to customize the administrative password to something other than "admin123", and you might want to override the default directories that Nexus uses to store repository data. To do this, log in as the administrative user and click on Server under Administration in the left-hand navigation menu as visible in Figure 6.1.



Figure 6.1: Administration Menu in the Left Hand Panel

The server configuration screens subsections are documented in the following.

6.1.1 SMTP Settings

Nexus sends email to users who need to recover user names and password. To do this, you'll need to configure the SMTP server settings in this dialog. This section of the form takes an SMTP Host and Port as well as other parameters relating to SMTP authentication and encryption. You can also change the From: header of an email from Nexus.

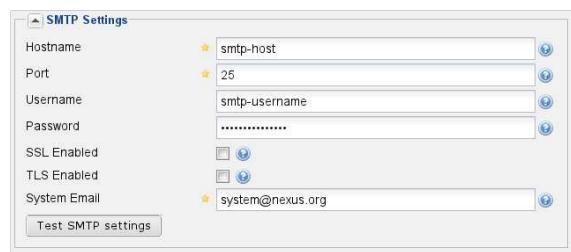


Figure 6.2: Administration SMTP Settings

6.1.2 HTTP Request Settings

The HTTP Request Settings allow you to configure the identifier which Nexus uses when it is making an HTTP request. You may want to change this if Nexus needs to use an HTTP Proxy, and the Proxy will only work if the User Agent is set to a specific value.

You can also add extra parameters to place on a GET request to a remote repository. You could use this to add identifying information to requests.

The amount of time Nexus will wait for a request to succeed when interacting with an external, remote repository can be configured with the Request Timeout and Request Retry Attempts settings.

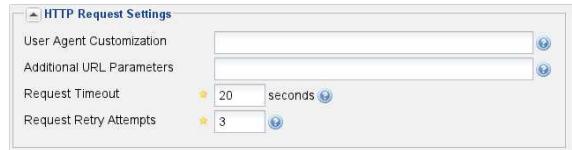


Figure 6.3: Administration HTTP Request Settings

6.1.3 Security Settings

You can choose to enable or disable security, enable or disable anonymous access, and set the username and password for anonymous access. If you choose to enable security, you are telling Nexus to enforce role-based access control to enforce read and write access to repositories.

The anonymous username and password is used to integrate with other realms that may need a special username for anonymous access. In other words, the username and password here is what we attempt to authorize when someone makes an anonymous request. You would change the anonymous username to "guest" if you wanted to integrate Nexus with Microsoft's Active Directory.

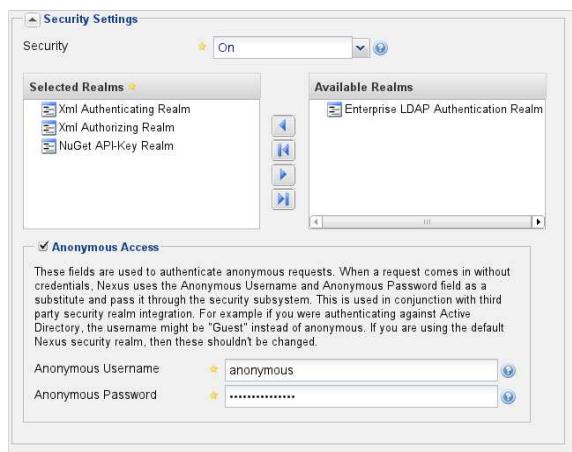


Figure 6.4: Administration Security Settings

6.1.4 Error Reporting Settings

Nexus can be configured to automatically file exception and error reports with the Nexus project in the Sonatype issue tracker. Activating this setting in your own Nexus installation helps to improve Nexus as the development team will receive automatic error reports if your Nexus instance experiences an error or a failure. The Nexus Server configuration's Automated Error Reporting Settings section is shown in Figure 6.5. This section accepts a JIRA username and password, and allows you to configure Nexus to use the default HTTP Proxy Settings when Nexus attempts to file an error report with the Sonatype issue tracker.

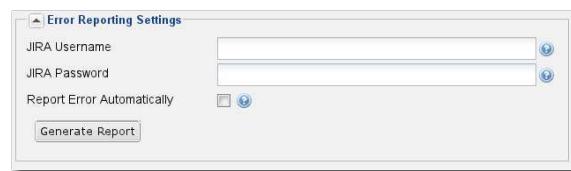


Figure 6.5: Administration Error Reporting Settings

To sign up for an account on the Sonatype JIRA instance, go to <http://issues.sonatype.org>. Once you see the web site shown in Figure 6.6, click on the "Signup" link below the Login form.

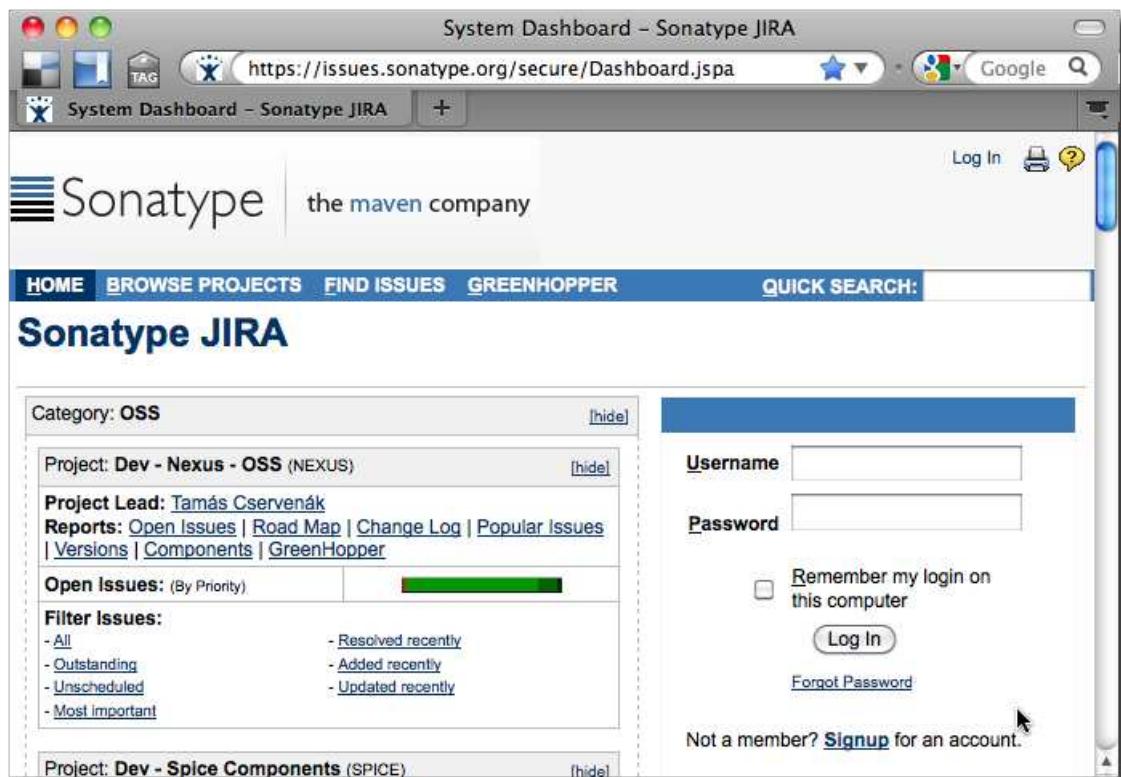


Figure 6.6: Sonatype Issue Tracker

Fill out the sign-up form shown in Figure 6.7, and choose a username and password. This is the username and password you should use in the Automated Error Reporting Settings section of the Server configuration shown in Figure 6.5.

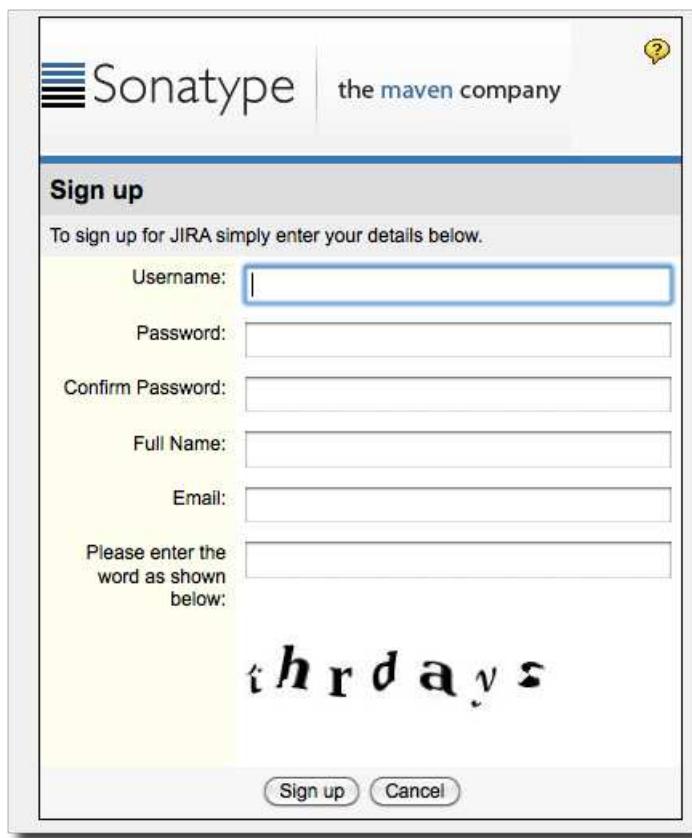


Figure 6.7: Signing Up for a Sonatype Issue Tracker Account

6.1.5 Application Server Settings

This section allows you to change the Base URL for your Nexus installation. It is used when generating links in emails and RSS feeds. The Sonatype Nexus repository is available on <http://repository.sonatype.org>, and it makes use of this Base URL field to ensure that links in emails and RSS feeds point to the correct URL. If you are hosting Nexus behind a proxy server and you want to make sure that Nexus always uses the specified Base URL, check the "Force Base URL" checkbox. If the Force Base URL is not checked, Nexus will craft URLs in HTTP responses based on the request URL, but it will use the Base URL when it is generating emails.

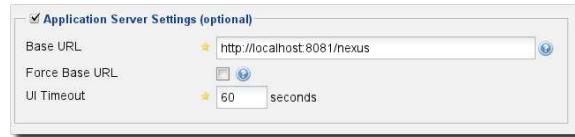


Figure 6.8: Administration Application Server Settings

6.1.6 Default HTTP Proxy Settings

If your Nexus instance needs to reach public repositories like the Central Repository via a proxy server, you can configure one or multiple servers in this settings section.

There are a number of HTTP Proxy settings for Nexus installations which need to be configured to use an HTTP Proxy. You can specify a host, port, and a number of authentication options which might be required by your proxy server.

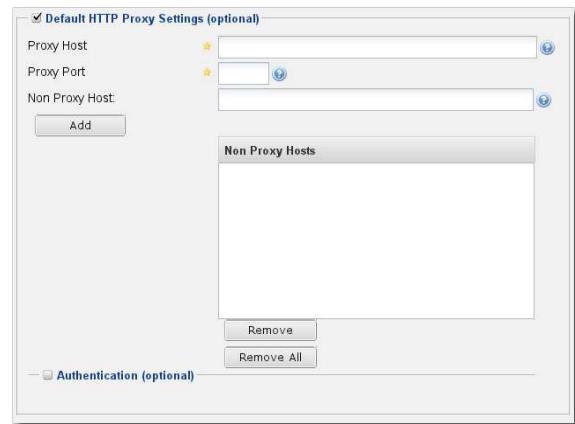


Figure 6.9: Administration Default HTTP Proxy Settings

6.1.7 System Notification Settings

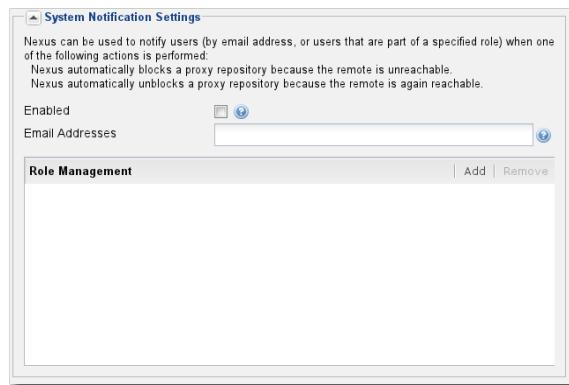


Figure 6.10: Administration System Notification Settings

6.1.8 New Version Availability

Nexus can notify you of new versions of Nexus via the Nexus interface. To enable this feature, check the Enable checkbox in the New Version Notification section of the Nexus server settings as shown in Figure 6.11.

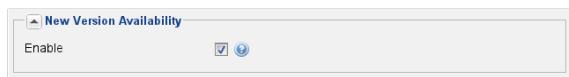


Figure 6.11: Administration New Version Availability

6.1.9 PGP Key Server Information

Nexus Professional uses a PGP Key Server to retrieve PGP keys when validating artifact signatures. To add a new Key Server URL, enter the URL in the Key Server URL field and click on the Add button. To remove a Key Server URL, click on the URL you wish to remove from the list and click on the Remove button. Key Servers are consulted in the order that they are listed in the Key Server URLs list, to reorder your Key Server URLs, click and drag a URL in the Key Server URLs list.



Figure 6.12: Administration PGP Key Server Information

6.2 Managing Repositories

To manage Nexus repositories, log in as the administrative user and click on Repositories in the Views/Repositories menu in the left-hand navigation menu.

Nexus provides for three different kinds of repositories - Proxy Repositories, Hosted Repositories and Virtual Repositories.

6.2.1 Proxy Repository

A proxy repository is a proxy of a remote repository. By default, Nexus ships with the following configured proxy repositories:

Apache Snapshots

This repository contains snapshot releases from the Apache Software Foundation <http://people.apache.org/repo/m2-snapshot-repository>

Codehaus Snapshots

This repository contains snapshot released from Codehaus <http://snapshots.repository.codehaus.org/>

Maven Central Repository

This is the central repository (for releases). <http://repo1.maven.org/maven2/>

6.2.2 Hosted Repository

A hosted repository is a repository which is hosted by Nexus. Maven ships with the following configured hosted repositories:

3rd Party

This hosted repository should be used for third-party dependencies not available in the public Maven repositories. Examples of these dependencies could be commercial, proprietary libraries such as an Oracle JDBC driver that may be referenced by your organization.

Releases

This hosted repository is where your organization will publish internal releases.

Snapshots

This hosted repository is where your organization will publish internal snapshots.

6.2.3 Virtual Repository

This serves as an adaptor to and from different types of repositories. Currently Nexus supports conversion to and from Maven 1 repositories and Maven 2 repositories and from Maven 2 to NuGet repositories.

6.2.4 Configuring Repositories

The Repositories window displayed in Figure 6.13 allows you to create, update and delete different repositories with the Add, Delete and Trash button. Use the Refresh button to update the displayed list of repositories and repository groups.

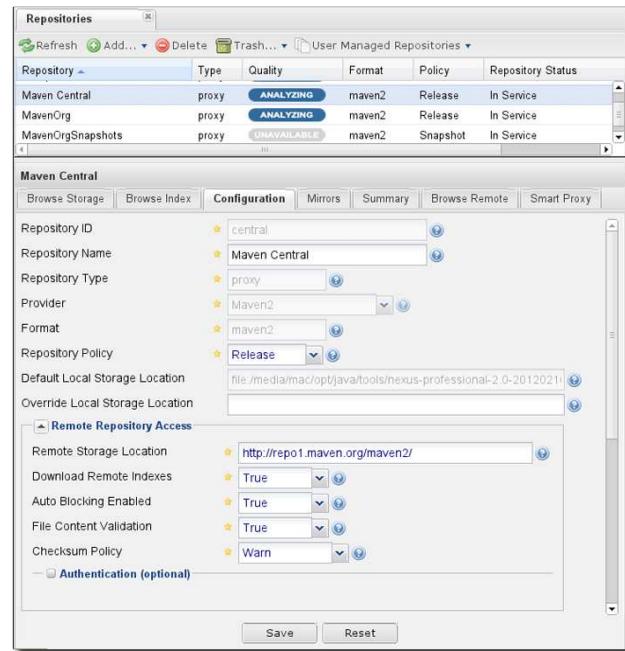


Figure 6.13: Repository Configuration Screen for a Proxy Repository

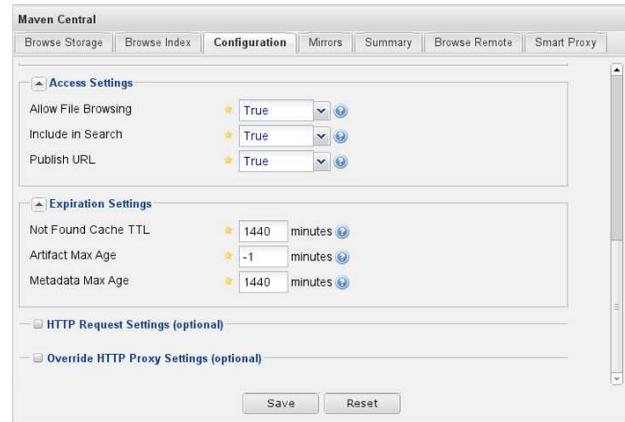


Figure 6.14: Repository Configuration Screen for a Proxy Repository

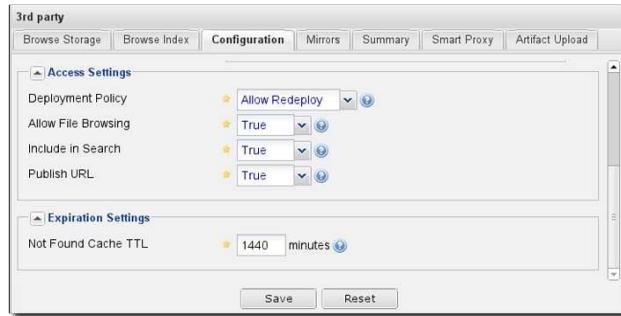


Figure 6.15: Proxy Configuration Access Settings for a Hosted Repository

Figure 6.13 and Figure 6.14 show the Repository configuration screen for a Proxy repository in Nexus. From this screen, you can manage the settings for proxying an external repository:

Repository ID

The repository ID is the identifier which will be used in the Nexus URL. For example, the central proxy repository has an ID of "central", this means that maven can access the repository directly at <http://localhost:8081/nexus/content/repositories/central>. The Repository ID must be unique in a given Nexus installation. ID is required.

Repository Name

The display name for a repository. Name is required.

Repository Type

The type of repository (proxy, hosted, or virtual). You can't change the type of a repository, it is selected when you create a repository.

Provider and Format

Provider and Format define in what format Nexus exposes the repository to external tools. Supported formats are Maven 1, Maven 2, NuGet, OBR, P2 and P2 Update Site.

Repository Policy

If a proxy repository has a policy of release than it will only access released versions from the remote repository. If a proxy repository has a policy of snapshot, it will download snapshots from the remote repository.

Default Storage Location

Not editable, shown for reference. This is the default storage location for the local cached contents of the repository.

Override Storage Location

You can choose to override the storage location for a specific repository. You would do this if

you were concerned about storage and wanted to put the contents of a specific repository (such as central) in a different location.

Remote Repository Access

This section tells Nexus where to look for and how to interact with the remote Maven repository being proxied.

Remote Storage Location

This is the URL of the remote Maven repository.

Download Remote Indexes

This field controls the downloading of the remote indexes. Currently only central has an index at <http://repo1.maven.org/maven2/.index>. If enabled, Nexus will download the index and use that for its searches as well as serve that up to any clients which ask for the index (like m2eclipse). The default for new proxy repositories is enabled, but all of the default repositories included in Nexus have this option disabled. To change this setting for one of the proxy repositories that ship with Nexus, change the option, save the repository, and then re-index the repository. Once this is done, artifact search will return every artifact available on the Maven Central repository.

Auto Blocking Enabled

If Auto blocking active is set to true, Nexus will automatically block a proxy repository if the remote repository becomes unavailable. While a proxy repository is blocked, artifacts will still be served to clients from a local cache, but Nexus will not attempt to locate an artifact in a remote repository. Nexus will periodically retest the remote repository and unblock the repository once it becomes available.

File Content Validation

If set to true, Nexus will perform a lightweight check on the content of downloaded files. This will prevent invalid content to be stored and proxied by Nexus, which otherwise can happen in cases where the remote repository (or some proxy between Nexus and the remote repository) for example returns an HTML page instead of the requested file.

Checksum Policy

Sets the checksum policy for a remote repository. This option is set to Warn by default. The possible values of this setting are:

- Ignore - Ignore the checksums entirely
- Warn - Print a warning in the log if a checksum is not correct
- StrictIfExists - Refuse to cache an artifact if the calculated checksum is inconsistent with a checksum in the repository. Only perform this check if the checksum file is present.
- Strict - Refuse to cache an artifact if the calculated checksum is inconsistent or if there is no checksum for an artifact.

Authentication

This section allows you to set a Username, Password, NT LAN Host, and NT Lan Manager Domain for a remote repository.

Access Settings

This section configures access settings for a repository.

Deployment Policy

This setting controls how a Hosted repository allows or disallows artifact deployment. If this 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 artifact once and any attempt to redeploy an artifact will result in an error. If this policy is set to "Allow Redeploy", clients can deploy artifacts to this repository and overwrite the same artifact in subsequent deployments. This option is visible for Hosted repositories as shown in Figure 6.15.

Allow File Browsing

When set to true, users can browse the contents of the repository with a web browser.

Include in Search

When set to true, this repository is search when you perform an Artifact Search in Nexus. If this setting is false, the contents of the repository are excluded from a search.

Publish URL

If this property is set to false, the repository will not be published on a URL, and you will not be able to access this repository remotely. You would set this configuration property to false if you want to prevent clients for connecting to this repository directly.

Expiration Settings

Nexus maintains a local cache of artifacts and metadata, you can configure expiration parameters for a proxy repository. The expiration settings are:

Not Found Cache TTL

If Nexus fails to locate an artifact, it will cache this result for a given number of minutes. In other words, if Nexus can't find an artifact in a remote repository, it will not repeated attempt to resolve this artifact until the Not Found Cache TTL time has been exceeded. The default for this setting is 1440 minutes (or 24 hours).

Artifact Max Age

Tells Nexus when that maximum age of an artifact is before it retrieves a new version from the remote repository. The default for this setting is -1 for a repository with a Release policy and 1440 for a repository with Snapshot policy.

Metadata Max Age

Nexus retrieves metadata from the remote repository. It will only retrieve updates to metadata after the Metadata Max Age has been exceeded. The default value for this setting is 1440 minutes (or 24 hours).

HTTP Request Settings

This section lets you change the properties of the HTTP request to the remote repository. In this section you can configure the User Agent of the request, add parameters to a request, and set the timeout and retry behaviour. This section refers to the HTTP request made from Nexus to the remote Maven repository being proxied.

Override HTTP Proxy Settings

This section lets you configure the HTTP Proxy for the request from Nexus to the remote repository. You can configure a proxy host and port plus an authentication settings you need tell Nexus to use an HTTP Proxy for all requests to a remote repository.

6.2.5 Selecting Mirrors for Proxy Repositories

Nexus also allows you to select which mirrors Nexus will consult for a particular Proxy repository. Clicking on the Mirrors tab will show the figure shown in Figure 6.16.

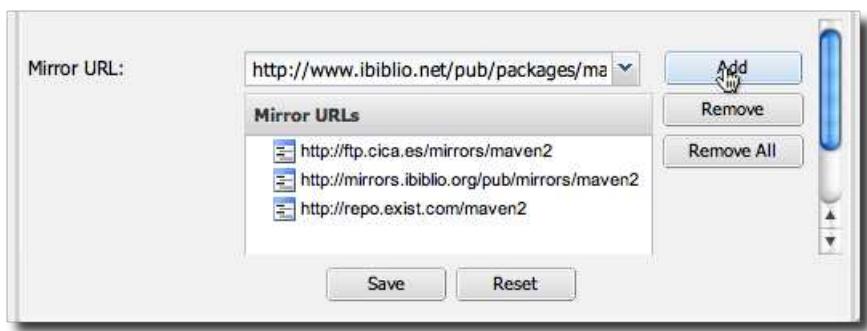


Figure 6.16: Configuring Mirrors for Proxy Repositories

To configure a mirror repository, click on the Mirror URL drop-down and select a mirror for the Proxy repository. Click the Add button, and Nexus will then be configured to download artifacts from the selected mirror. Nexus will always download checksums and metadata from the original (or Canonical) URL for a proxy repository. For example, if Nexus is going to download an artifact, it will retrieve the MD5 checksum from the original Maven Central repository and then retrieve the artifact from the selected mirror.

6.2.6 Adding a Mirror Entry for a Hosted Repository

If you are logged in as a user with Administrative privilege, there will be a Mirrors tab available when you are viewing a Hosted repository, clicking on this Mirrors tab will show the form shown in Figure 6.16. This tab contains a list of mirror URLs for this hosted repository. If there are other sites which mirror the contents of this hosted repository, this tab allows you to populate the repository mirror metadata with those URLs.

This repository mirror metadata can then be consumed by other systems that interact with your hosted repository. For example, if you have a release repository which is used by your customers or by the general public, if one of people consuming your Hosted repository is also running a Nexus, they can configure a Proxy repository that targets your Hosted repository and they can use the mirror metadata to

configure their instance of Nexus to consume artifacts from mirrors of your Hosted repository.

6.2.7 Viewing Repository Summary Panel

The Repository Summary panel can be loaded by selecting a Hosted, Proxy, or Virtual repository and then clicking on the Summary tab. When viewing the Summary tab of a Hosted repository, as shown in Figure 6.17, you will also see the Distribution Management settings which can be used to configure Maven to publish artifacts to a Hosted repository.

The screenshot shows the Nexus web interface with the 'Repositories' tab selected. A table lists several repositories:

Repository	Type	Quality	Format	Policy	Repository Status
NuGet Public Group	group	UNAVAILABLE	nuget		
OurPublicAPIs	group	UNAVAILABLE	maven2		
Public Repositories	group	UNAVAILABLE	maven2		
3rd party	hosted	UNAVAILABLE	maven2	Release	In Service
Apache Snapshots	proxy	UNAVAILABLE	maven2	Snapshot	In Service

The '3rd party' repository is selected. Below the table, tabs for 'Browse Storage', 'Browse Index', 'Configuration', 'Mirrors', 'Summary' (which is active), 'Smart Proxy', and 'Artifact Upload' are visible. The 'Repository Information' section displays details for the '3rd party' repository, including its ID, name, type, policy, format, and group membership. The 'Distribution Management' section shows the XML configuration for publishing artifacts to this repository:

```
<distributionManagement>
<repository>
<id>thirdparty</id>
<url>http://localhost:8081/nexus/content/repositories/thirdparty</url>
</repository>
</distributionManagement>
```

Figure 6.17: Repository Summary Panel for a Hosted Repository

The Repository Summary panel for a Proxy repository, as shown in Figure 6.18, contains all of the repository identifiers and configuration in addition to the size of the local storage for the proxy repository and the URL of the remote repository.

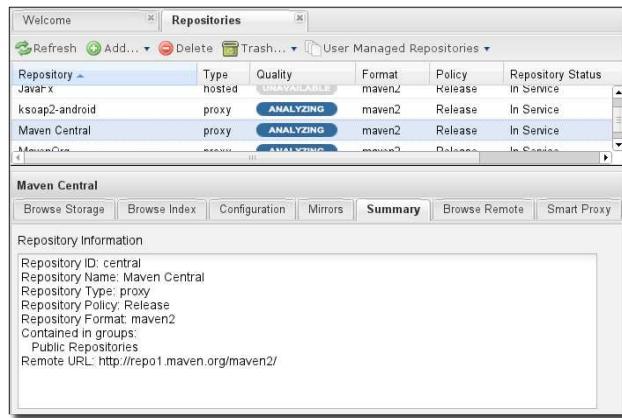


Figure 6.18: Repository Summary Panel for a Proxy Repository

The Repository Summary panel for a Virtual repository, as shown in Figure 6.19, displays repository identifiers and the size of the Virtual repository on disk.

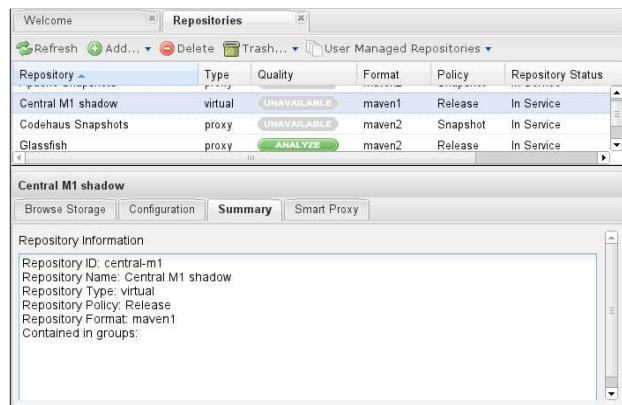


Figure 6.19: Repository Summary Panel for a Virtual Repository

6.2.8 Auto Block/Unblock of Remote Repositories

What happens when Nexus is unable to reach a remote repository? If you've defined a proxy repository, and the remote repository is unavailable Nexus will now automatically block the remote repository. Once a repository has been auto-blocked, Nexus will then periodically retest the remote repository and unblock the repository once it becomes available. You can control this behaviour by changing the Auto-blocking Active setting under the Remote Repository Access section of the proxy repository configuration as shown in the following figure:



Figure 6.20: Configuring Remote Repository Auto Block/Unblock

6.3 Managing Groups

Groups are a powerful feature of Nexus. They allow you to combine multiple repositories and other repository groups in a single URL. Use the left hand panel **Repositories** menu item in the **Views/Repositories** menu to access the repositories and groups management interface.

Nexus ships with one group: `public`. The `Public Repositories` group combines the multiple important external proxy repositories like the Central Repository with the hosted repositories: 3rd Party, Releases, and Snapshots.

In Section 4.2 we configured Maven via the `settings.xml` to look for artifacts in the `public` group managed by Nexus. Figure 6.21 shows the group configuration screen in Nexus, in this figure you can see the contents of the `public`

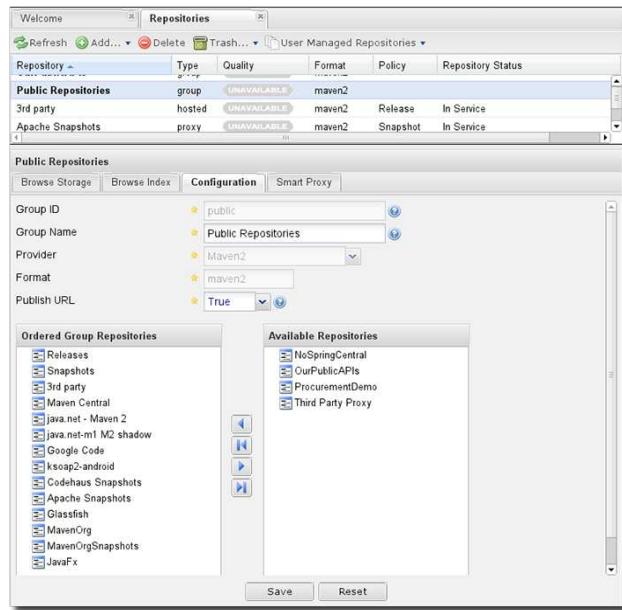


Figure 6.21: Group Configuration Screen in Nexus

Note that the order of the repositories listed in Order Group Repositories is important. When Nexus searches for an artifact in a group it will return the first match. To reorder a repository in this list, click and drag the repositories and groups in the Ordered Group Repositories selection list.

The order of repositories or other groups in a group can be used to influence the effective metadata that will be retrieved by Maven from a Nexus Repository Group. We recommend placing release repositories higher in the list than snapshot repositories so that LATEST and RELEASE versions are merged appropriately.

We also recommend placing repositories with a higher probability of matching the majority of artifacts higher in this list. If most of your artifacts are going to be retrieved from the Maven Central Repository, putting Central higher in this list than a smaller, more focused repository is going to be better for performance as Nexus is not going to interrogate the smaller remote repository for as many missing artifacts.

6.4 Managing Routes

Nexus Routes are like filters you can apply to Nexus Groups, they allow you to configure Nexus to include or exclude repositories from a particular artifact search when Nexus is trying to locate an artifact in a repository group. There are a number of different scenarios in which you might configure a route in Nexus.

The most common is when you want to make sure that you are retrieving artifacts in a particular group ID from a particular repository. This is especially useful when you want to make sure that you are trying to retrieve your own organization's artifacts from the hosted Release and Snapshot repositories.

Routes are applicable when you are trying to resolve an artifact from a repository group; using Routes allow you to modify the repositories Nexus will consult when it tries to resolve an artifact from a group of repositories.

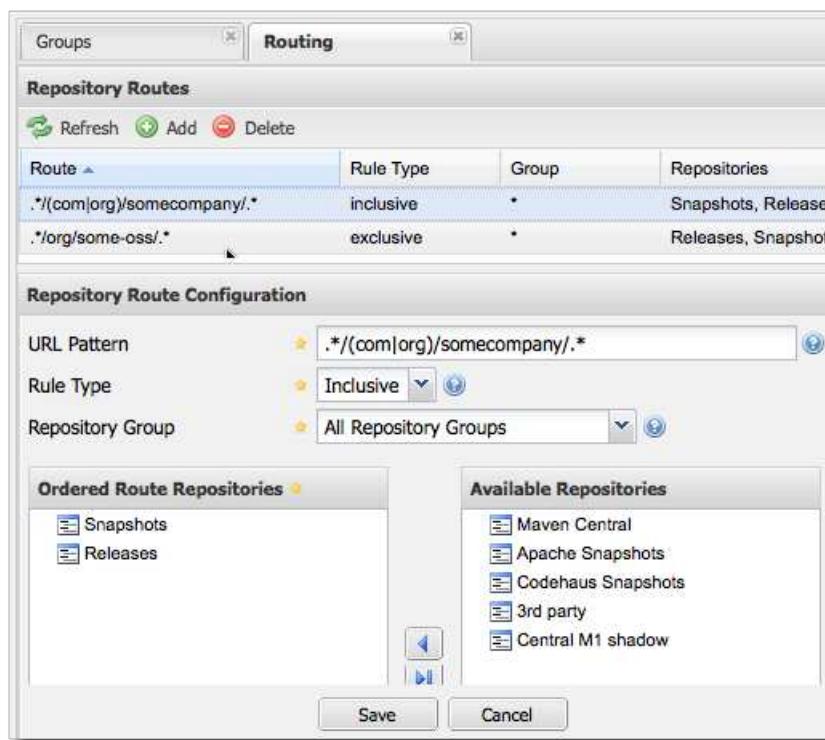


Figure 6.22: Routes Configuration Screen in Nexus

Figure 6.22 shows the Repository Route Configuration screen. Clicking on a route will bring up a screen which will allow you to configure the properties of a route. The configuration options available for a route are:

URL Pattern

This is the pattern which Nexus will use to match a request to Nexus. If the regular expression in this pattern is matched, Nexus will either include or exclude the listed repositories from a particular artifact query. In Figure 6.22 the two patterns are:

".*/(com|org)/somecompany/*"

This pattern would match all paths which includes either "/com/somecompany/" or "/org/somecompany/". The expression in the parenthesis matches either com or org, and the "*" matches zero or more characters. You would use a route like this to match your own organization's artifacts and map these requests to the hosted Nexus Releases and Snapshots repositories.

".*/org/some-oss/*"

This pattern is used in an exclusive route. It matches every path that contains "/org/some-oss/". This particular exclusive route excludes the local hosted Releases and Snapshots directory for all artifacts which match this path. When Nexus tries to resolve artifacts that match this path, it will exclude the Releases and Snapshots repositories.

Rule Type

Rule Type can be either "inclusive" or "exclusive". An inclusive rule type defines the set of repositories which should be searched for artifacts when the URL pattern has been matched. An exclusive rule type defines repositories which should not be searched for a particular artifact.

Ordered Route Repositories

This is an ordered list of repositories which Nexus will search to locate a particular artifact. Nexus searches top to bottom; if it's looking for an artifact, it will return the first match. When Nexus is looking for metadata, all repositories in a group are checked and the results are merged. The merging is applied giving preference to the earlier repositories. This is relevant when a project is looking for a LATEST or a RELEASE version. Within a Nexus Group, you should define the release repositories before the snapshot repositories, otherwise LATEST may incorrectly resolve to a snapshot version.

In this figure you can see the two dummy Routes that Nexus has as default routes.

The first route is an inclusive route, it is provided as an example of a custom route an organization might use to make sure that internally generated artifacts are resolved from the Releases and Snapshots repositories. If your organization's group IDs all start with com.somecompany, and if you deploy internally generated artifacts to the Releases and Snapshots repositories, this Route will make sure that Nexus doesn't waste time trying to resolve these artifacts from public Maven repositories like the Maven Central Repository or the Apache Snapshots repository.

The second dummy route is an exclusive route. This route excludes the Releases and Snapshots repositories when the request path contains "/org/some-oss". This example might make more sense if we replaced "some-oss" with "apache" or "codehaus". If the pattern was "/org/apache", this rule is telling Nexus to exclude the internal Releases and Snapshots repositories when it is trying to resolve these dependencies. In other words, don't bother looking for an Apache dependency in your organization's internal repositories.

What if there is a conflict between two routes? Nexus will process inclusive routes before it will process the exclusive routes. Remember that Nexus Routes only affect Nexus' resolution of artifacts when it is searching a Group. When Nexus starts to resolve an artifact from a Nexus Group it will start with the list of repositories in a group. If there are matching inclusive routes, Nexus will then take the intersection of the repositories in the Group and the repositories in the inclusive Nexus Route. The order as defined in the Nexus Group will not be affected by the Inclusive route. Nexus will then take the result of applying the inclusive route and apply the exclusive route to that list of repositories. The resulting list is then searched for a matching artifact.

One straightforward use of routes is to create a route that excludes the Maven Central repository from all searches for your own organization's hosted artifacts. If you are deploying your own artifacts to Nexus under a groupId of org.mycompany, and if you are not deploying these artifacts to a public repository, you can create a rule that tells Nexus not to interrogate Central for your own organization's artifacts. This will improve performance because Nexus will not need to communicate with a remote repository when it serves your own organization's artifacts. In addition to the performance benefits, excluding Central from searches for your own artifacts will reduce needless queries to the public repositories.

To summarize, there are creative possibilities with Routes that the designers of Nexus may not have anticipated, but we advise you to proceed with caution if you start relying on conflicting or overlapping Routes. Use Routes sparingly, and use course URL patterns, as Nexus evolves there will be more features which allow for more fine grained rules to allow you to prohibit requests for specific artifacts and specific versions of artifacts. Remember that routes are only applied to Nexus Groups, routes are not used when an artifact is requested from a specific repository.

6.5 Managing Scheduled Tasks

Nexus allows you to schedule tasks that will be applied to all repositories or to specific repositories on a configurable schedule. Use the Scheduled Tasks menu item in the Administration menu visible in Figure 6.1 to access the screen shown in Figure 6.23, that allows you to manage your Scheduled Tasks.

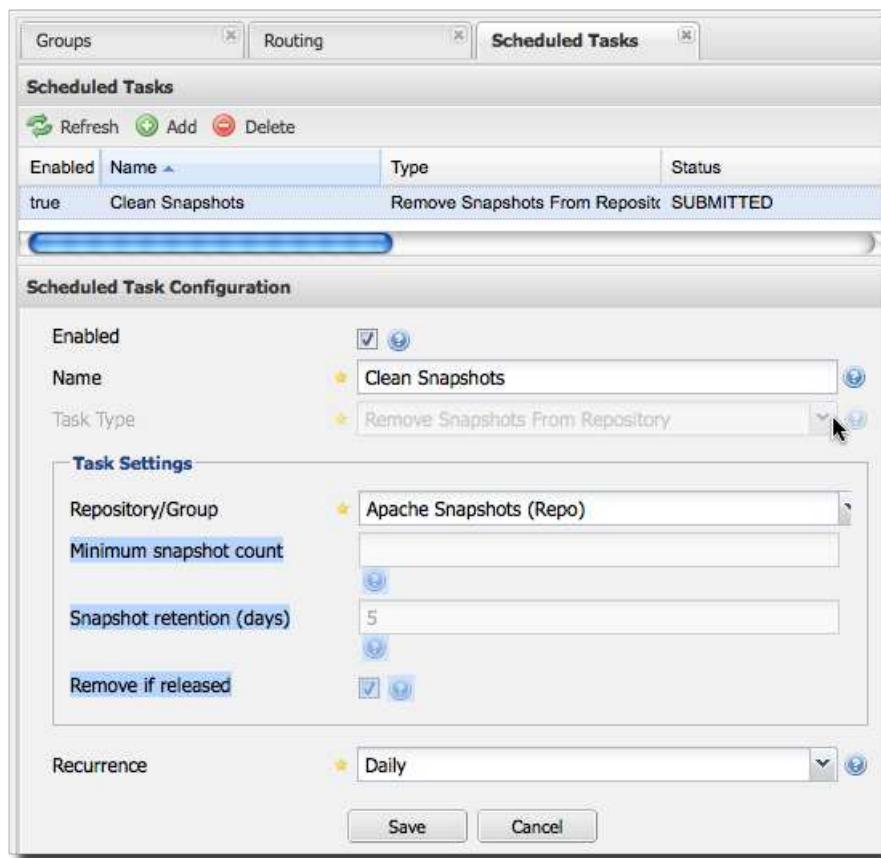


Figure 6.23: Managing Nexus Scheduled Tasks

You can create the following kinds of scheduled tasks:

Download Indexes

This scheduled task will cause Nexus to download indexes from remote repositories for proxied repositories. The Download Remote Indexes configuration also needs to be enabled on the proxy repository.

Empty Trash

The Evict and Purge actions do not delete data from the Nexus working directory. They simply move data to be cleared or evicted to a trash directory under the Nexus work directory. This service deletes the data in this trash directory.

Evict Unused Proxied Items From Repository Caches

Use it or lose it. This scheduled task tells Nexus to get rid of all proxied items which haven't been "used" (referenced or retrieved by a client). This can be a good job to run if you are trying to conserve storage space. In this service you can specify the number of days over which Nexus will look for activity before making the decision to evict an artifact. (See note about deletion.)

Optimize Repository Index

To speed up searches in Nexus, this task tells the internal search engine to optimize its index files. This has no affect on the indexes published by Nexus. Typically, this task does not have to run more than once a week.

Publish Indexes

Just as Maven downloads an index from a remote repository, Nexus can publish an index in the same format. This will make it easier for people using m2eclipse or Nexus to interact with your repositories.

Purge Nexus Timeline

Nexus maintains a lot of data that relates to the interaction between itself, proxied remote repositories, and clients on Nexus. While this information can be important for purposes of auditing, it can also take up storage space. Using this scheduled task you can tell Nexus to periodically purge this information. (See note about deletion.)

Repair Repositories Index

In certain cases it might be required to remove the internal index as well as the published ones of a repository. This task does that and then rebuilds the internal index by first trying to download remote indexes (if a proxy repository), then scanning the local storage and updating the internal index accordingly. Lastly, the index is published for the repository as well. There should be no need to schedule this task. But when upgrading Nexus, the upgrade instructions may sometimes include a manual step of executing this task.

Remove Snapshots from Repository

Often, you will want to remove snapshots from a snapshot repository to preserve storage space. Note that configuring and running this job is not enough to reclaim disk space. You will also need to configure a scheduled job to empty the trash folder. Files are not deleted by the Remove Snapshots job, they are only moved into the Trash folder. When you create a scheduled task to remove snapshots, you can specify:

Minimum Snapshots to preserve in a repository - This configuration option allows you to specify a minimum number of SNAPSHOTs to preserve per artifact. For example, if you configured this option with a value of 2, Nexus will always preserve at least two SNAPSHOT artifacts.

Snapshot Retention (in days) - This configuration option allows you to specify the number of days to retain SNAPSHOT artifacts. For example, if you want to make sure that you are always keeping the last three day's worth of SNAPSHOT artifacts, configure this option with a value of 3.

Whether snapshots should be removed if an artifact has been released - If your Nexus repository also contains a release repository, you can configure Nexus to remove all SNAPSHOT artifacts once an artifact has been released.

Synchronize Shadow Repository

This service synchronizes a shadow (or virtual) repository with its master repository.

Update Repositories Index

If files are deployed directly to a repository's local storage (not deployed through Nexus), you will need to instruct Nexus to update its index. When executing this task, Nexus will update its index by first downloading remote indexes (if a proxy repository) and then scan the local storage to index the new files. Lastly, the index is published for the repository as well. Normally, there should be no need to schedule this task. One possible except would be if files are deployed directly to the local storage regularly.

Note

The Evict and Purge actions do not delete data from the Nexus working directory. They simply move data to be cleared or evicted to a trash directory under the Nexus work directory. If you want to reclaim disk space, you need to clear the Trash on the Browse Repositories screen. If something goes wrong with a evict or clear service, you can move the data back to the appropriate storage location from the trash. You can also schedule the Empty Trash service to clear this directory on a periodic basis.

When you create a new task you can configure it to apply to all repositories, the repositories in a Nexus Group, or a specific Nexus Repository. A task can be scheduled to run once at a specific date and time, or periodically once every Day, Week, or Month. If none of these options suit your specific needs, you can select a recurrence of "Advanced" which will allow you to supply your own cron expression to specify when the job should execute. There is also the recurrence option of "Manual", which will allow you to initiate an execution of the task manually.

Tip

In order to keep the heap usage in check it is recommended that you schedule an "optimize indexes" task to run weekly. An number of other maintenance tasks should also be scheduled for production deployments.

6.5.1 Managing Configuration Backups with a Scheduled Task

Nexus Professional includes a scheduled task which allows you to create automated, scheduled backups of your Nexus configuration. This scheduled job will archive the contents of the sonatype-work/nexus/conf directory. Figure 6.24 shows a scheduled configuration backup job configured to backup the contents of the Nexus configuration every day at 12:15 AM.

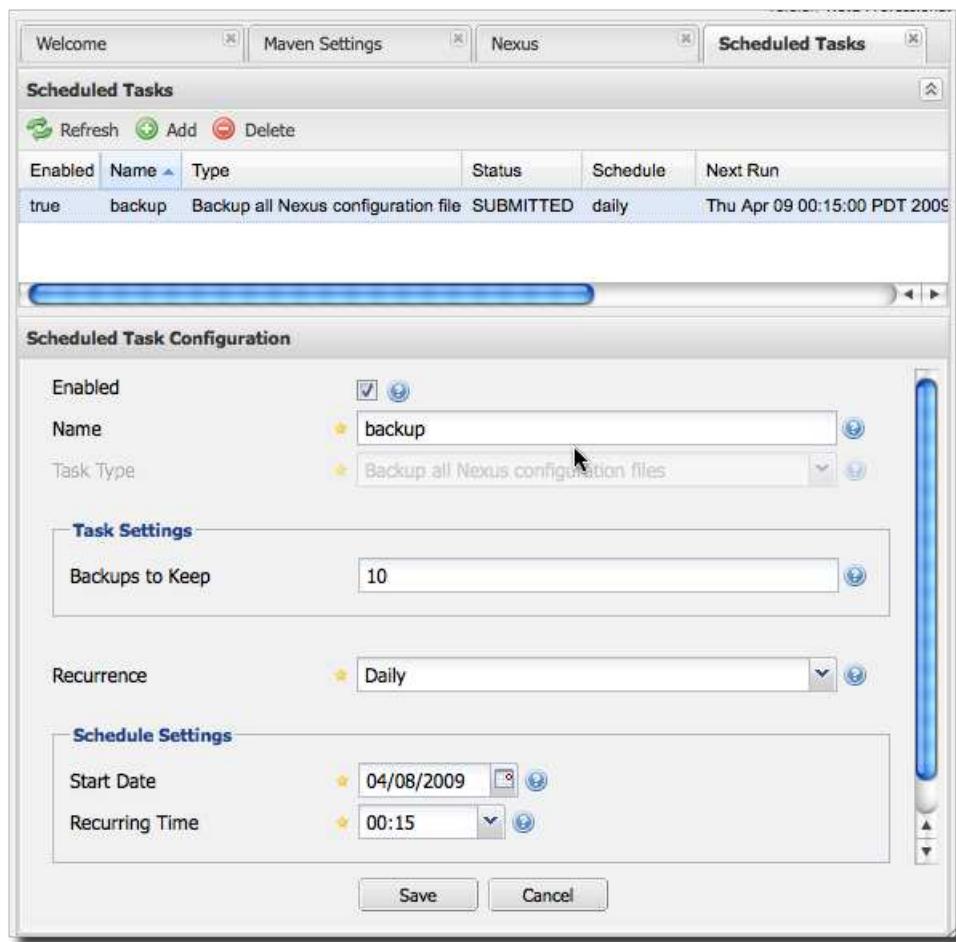


Figure 6.24: Configuring a Scheduled Backup of Nexus Configuration

Once a backup has been run, the contents of the backup will be available in `sonatype-work/nexus/backup` in a series of ZIP archives which include the date and a timestamp.

6.6 Managing Security

Nexus has role-based access control (RBAC) which gives administrators very fine-grained control over who can read from a repository (or a subset of repositories), who can administer the server, and who can deploy to repositories. The security model in Nexus is also so flexible as to allow you to specify that only certain users or roles can deploy and manage artifacts in a specific repository under a specific groupId or asset class. The default configuration of Nexus ships with four roles and four users with a standard set of permissions that will make sense for most users. As your security requirements evolve, you'll likely need to customize security settings to create protected repositories for multiple departments, or development groups. Nexus provides a security model which can adapt to any scenario. The Security configuration is done via menu items in the left hand Security menu.

Nexus' Role-based access control (RBAC) system is designed around the following four security concepts:

Privileges

Privileges are rights to read, update, create, or manage resources and perform operations. Nexus ships with a set of core privileges which cannot be modified, and you can create new privileges to allow for fine-grained targeting of role and user permissions for specific repositories.

Targets

Privileges are usually associated with resources or targets. In the case of Nexus, a target can be a specific repository or a set of repositories grouped in something called a repository target. A target can also be a subset of a repository or a specific asset classes within a repository. Using a target you can apply to a specific privilege to apply to a single groupId.

Roles

Collections of privileges can be grouped into roles to make it easier to define collections of privileges common to certain classes of users. For example, deployment 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. A role has one or more privilege and/or one or more roles.

Users

Users can be assigned roles and privileges, and model the individuals who will be logging into Nexus and read, deploying, or managing repositories.

6.7 Managing Privileges

Nexus has three types of privileges: application privileges which cover actions a user can execute in Nexus, repository target privileges which govern the level of access a user has to a particular repository or repository target, and repository view privileges which control whether a user can view a repository. Behind the scenes, a privilege is related to a single REST operation and method like create, update, delete, read.

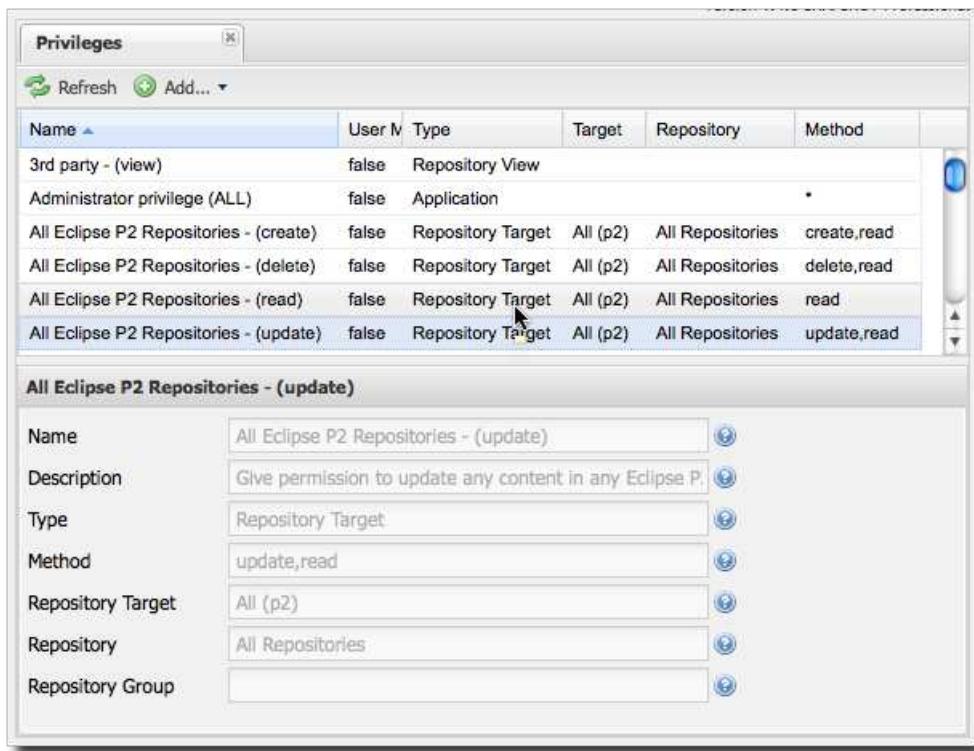


Figure 6.25: Managing Security Privileges

To create a new privilege, click on the Add... button in the Privileges panel and choose Repository Target privilege. Creating a privilege will load the New Repository Target Privilege form shown in Figure 6.26. This form takes a privilege name, a privilege description, the repository to target, and a repository target.

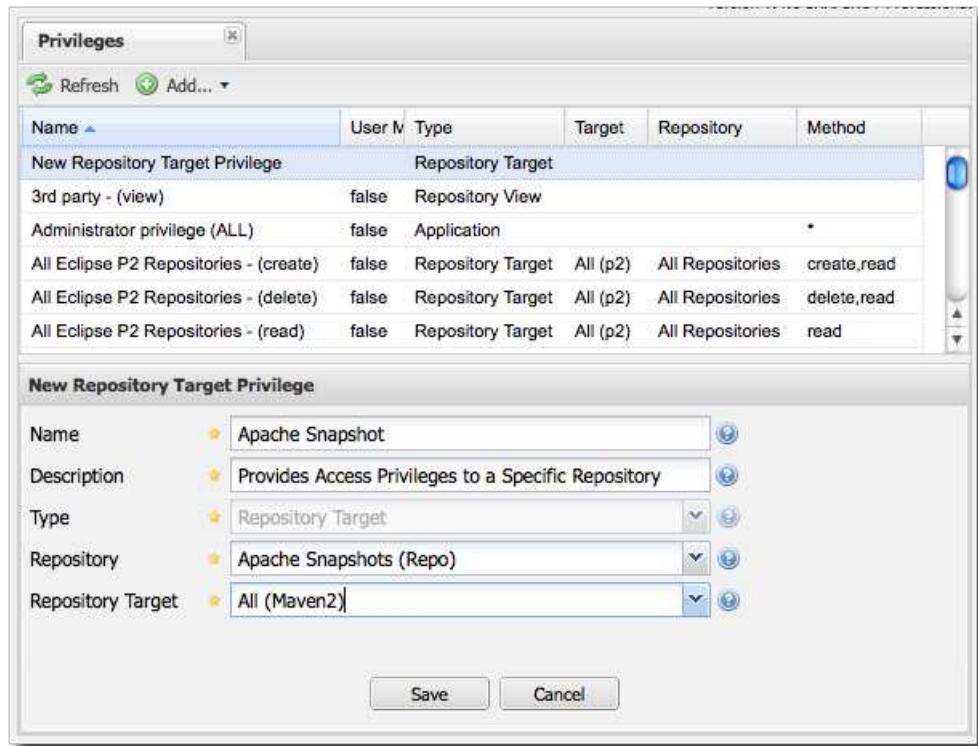


Figure 6.26: Managing Security Privileges

Once you create a new privilege, it will create four underlying privileges: create, delete, read, and update. The four privileges created by the form in Figure 6.26 are shown in Figure 6.27.

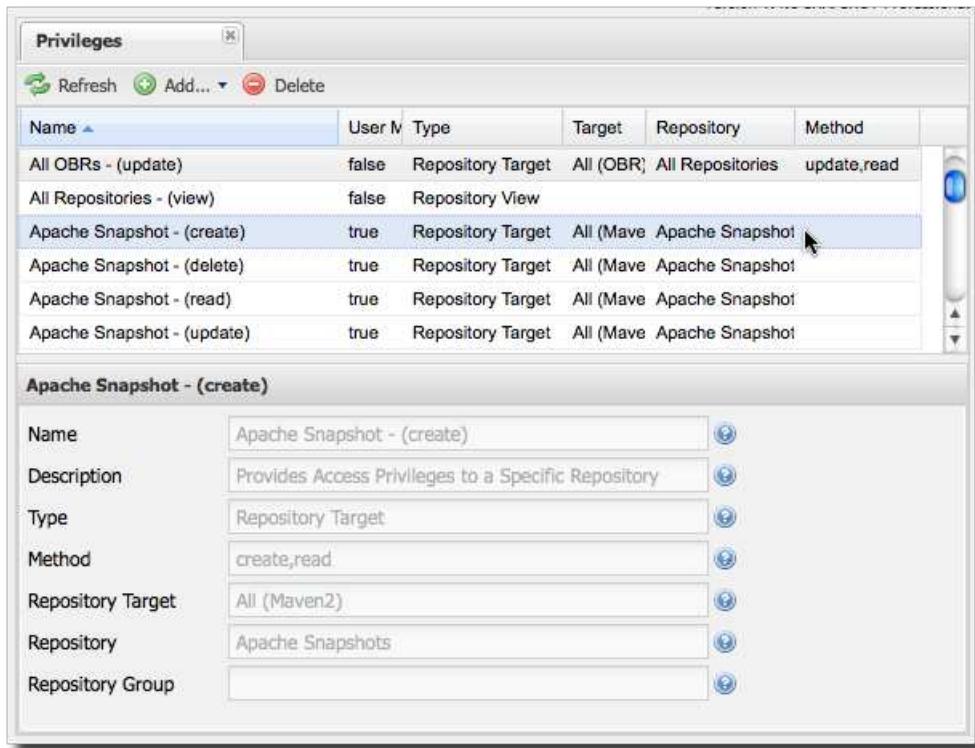


Figure 6.27: Create, Delete, Read, and Update Privileges Created

6.8 Managing Repository Targets

A target is a set of regular expressions to match on a path (in the same way as the route rules work). Access the management interface via the Repository Targets menu item in the left hand Views/Repositories menu.

Repository targets allow you to define for example a target called Apache Maven which is "/org/apache/maven/*". You can then add a new privilege that relates to the target and controls the CRUD operations for artifacts matching that path (the privilege can span multiple repositories if you want). You could thus delegate all control of org.apache.maven targets to a "Maven" team. In this way, you don't need to create separate repositories for each logical division of your artifacts.

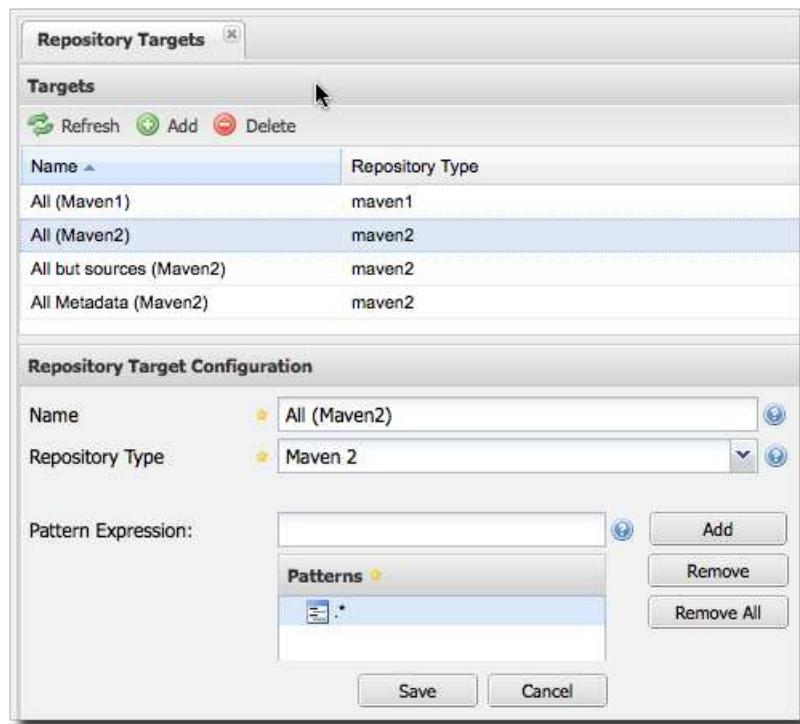


Figure 6.28: Managing Repository Targets

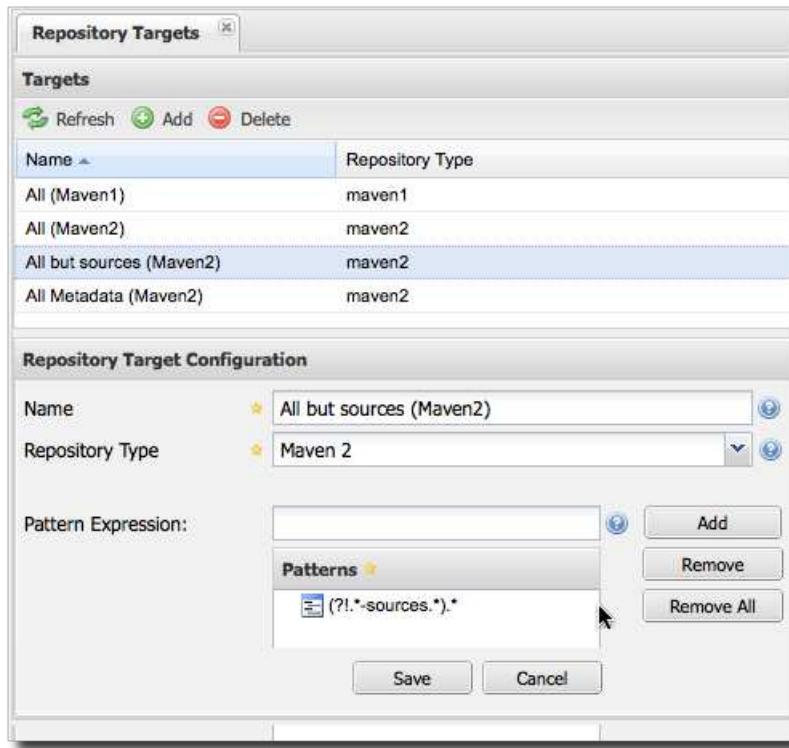


Figure 6.29: Managing Repository Targets

6.9 Managing Roles

Nexus ships with four roles: Nexus Administrator Role, Nexus Anonymous Role, Nexus Developer Role, and Nexus Deployment Role. Click on the Roles link under Security in the Nexus menu to show the list of roles shown in Figure 6.30.

Name	Mapping	Session Timeout	Description
Nexus Administrator Role		30	Administration role for Nexus
Nexus Anonymous Role		60	Anonymous role for Nexus
Nexus Deployment Role		60	Deployment role for Nexus
Nexus Developer Role		30	Developer role for Nexus
Repo: All Eclipse P2 Repositories (Full Control)		60	Gives access to create/read/update
Repo: All Eclipse P2 Repositories (Read Only)		60	Gives access to read ALL content
Repo: All Maven Site Repositories (Full Control)		60	Gives access to create/read/update
Repo: All Maven Site Repositories (Read Only)		60	Gives access to read ALL content
Repo: All OBRs (Full Control)		60	Gives access to create/read/update
Repo: All OBRs (Read Only)		60	Gives access to read ALL content
Repo: All Repositories (Full Control)		60	Gives access to create/read/update
Repo: All Repositories (Read Only)		60	Gives access to read ALL content
Staging: Deployer (admin)		60	Role that gives access to stage to

Figure 6.30: Viewing the List of Defined Roles

To create a new role, click on the Add... button and fill out the New Nexus Role form shown in Figure 6.31.

When creating a new role, you will need to supply a role identifier, a role name, a description, and a session timeout. Roles are comprised of other roles and individual privileges, to assign a role or privilege to a role, click on the role or privilege under Available Roles/Privileges and drag the role or privilege to the Selected Roles/Privileges list.

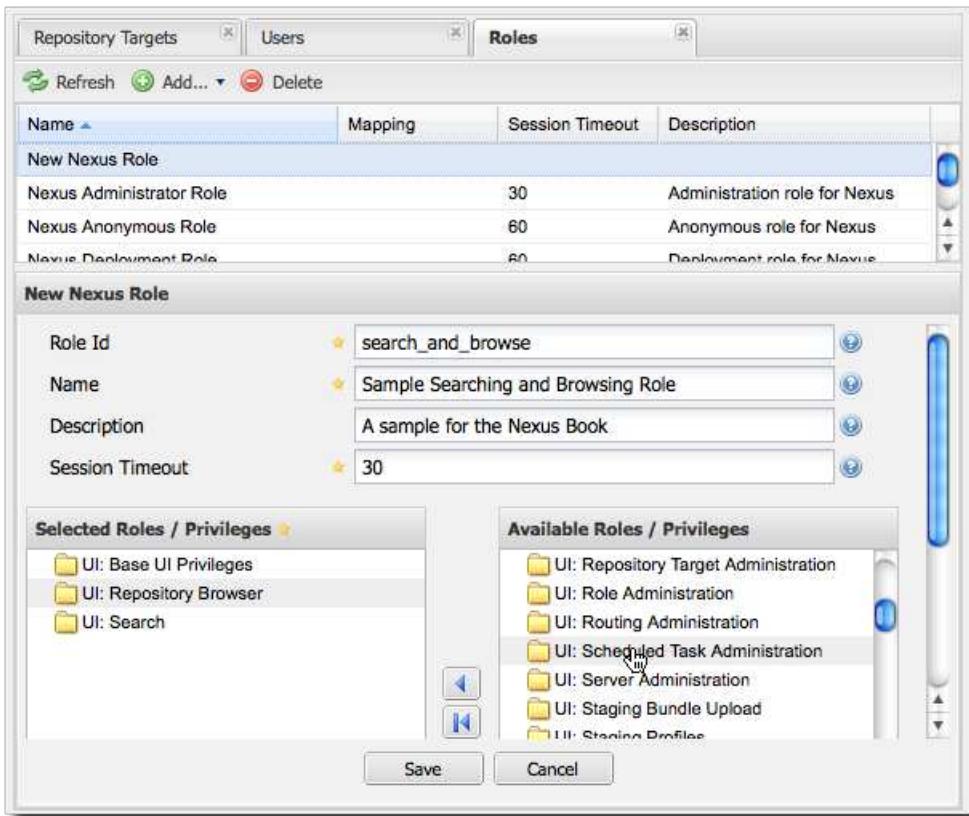


Figure 6.31: Creating a New Role

The built-in roles Nexus Administrator Role, Nexus Anonymous Role, Nexus Deployment Role, and Nexus Developer Role are managed by Nexus and can not be edited or deleted. Selecting one of these built-in roles will load the form shown in Figure 6.32.

The screenshot shows the Nexus Roles management interface. At the top, there are two tabs: 'Repository Targets' and 'Roles'. Below them is a toolbar with 'Refresh', 'Add...', and 'Delete' buttons. A table lists several roles:

Name	Mapping	Session Timeout	Description
Nexus Administrator Role		30	Administration role for Nexus
Nexus Anonymous Role		60	Anonymous role for Nexus
Nexus Deployment Role		60	Deployment role for Nexus
Nexus Developer Role		30	Developer role for Nexus

The 'Nexus Administrator Role' is selected, highlighted with a blue border. Below the table, a message box displays the following text:

This is an internal Nexus resource which cannot be edited or deleted.

Below the message box, there are four input fields with their current values:

- Role Id: admin
- Name: Nexus Administrator Role
- Description: Administration role for Nexus
- Session Timeout: 30

At the bottom, there are two panels: 'Selected Roles / Privileges' and 'Available Roles / Privileges'. The 'Selected Roles / Privileges' panel contains:

- Security administrator privilege (ALL)
- Administrator privilege (ALL)

The 'Available Roles / Privileges' panel contains:

- Nexus Anonymous Role
- Nexus Deployment Role

At the very bottom are 'Save' and 'Reset' buttons.

Figure 6.32: Viewing an Internal Role

A Nexus role is comprised of other Nexus roles and individual Nexus privileges. To view the component parts of a Nexus Role, select the role in the Roles panel and then choose the Role Tree tab as shown in Figure 6.33.

The screenshot shows the Nexus Repository Management interface with the 'Roles' tab selected. The main pane displays a table of roles with columns: Name, Mapping, Session Timeout, and Description. The 'Nexus Anonymous Role' is currently selected, as indicated by a blue selection bar at the top of its row. The 'Description' column for this role states 'Anonymous role for Nexus'. Below the table, a detailed view of the 'Nexus Anonymous Role' is shown in a tree structure under the 'Role Tree' tab. The tree includes categories like 'UI: Repository Browser', 'UI: Search', and 'UI: System Feeds', each containing specific permissions such as 'Read Repository Status', 'Repositories - (read)', 'Repository Groups - (read)', 'Checksum Search', 'Search Repositories', 'Artifact Download', and more.

Name	Mapping	Session Timeout	Description
Nexus Administrator Role	30	Administration role for Nexus	
Nexus Anonymous Role	60	Anonymous role for Nexus	
Nexus Deployment Role	60	Deployment role for Nexus	
Nexus Developer Role	30	Developer role for Nexus	

Figure 6.33: Managing Security Roles

With the Repository Targets, you have fine grained control over every action in the system. For example you could make a target that includes everything except sources `(.?!sources)\\.` and assign that to one group while giving yet another group access to everything. This means you can host your public and private artifacts in a single repository without giving up control of your private artifacts.

6.10 Managing Users

Nexus ships with three users: admin, anonymous, and deployment. The admin user has all privileges, the anonymous user has read-only privileges, and the deployment user can both read and deploy to repositories. If you need to create users with a more focused set of permissions, you can click on Users under Security in the left-hand navigation menu. Once you see the list of users, you can click on a user to edit that specific user's user ID, name, email, or status. You can also assign or revoke specific roles or permissions for a particular user.

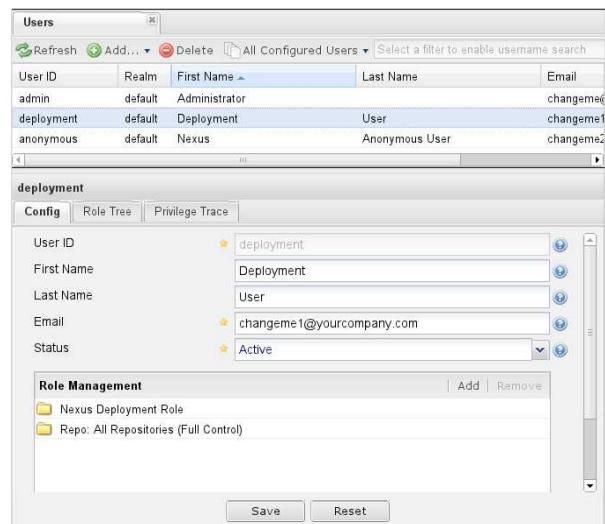


Figure 6.34: Managing Users

Clicking the Add button in the Role Management section will bring up the list of available roles in a pop up window visible in Figure 6.35. It allows you filter and search for roles and add one or multiple roles to the user.

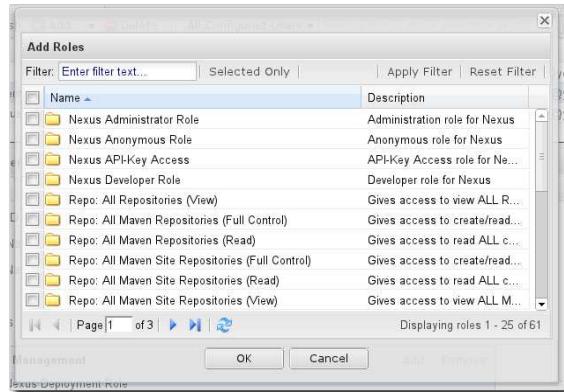


Figure 6.35: Adding Roles to a User

A user can be assigned one or more roles which in turn can include references to other Nexus roles or to individual Nexus privileges. To view a tree of assigned Nexus roles and privileges, select the Role Tree for a particular user as shown in Figure 6.36.

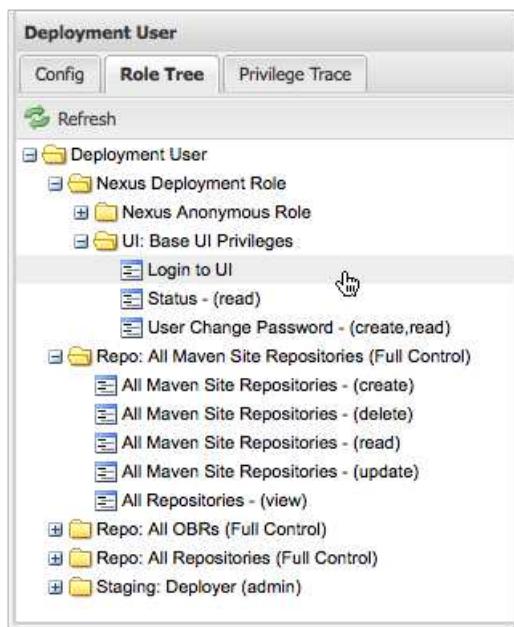


Figure 6.36: Nexus User Role Tree

If you need to find out exactly how a particular user has been granted a particular privilege, you can use the Privilege Trace pane as shown in Figure 6.37. The Privilege Trace pane lists all of the privileges that have been granted to a particular user. Clicking on a privilege loads a tree of roles that grant that particular privilege to a user. If a user has been assigned a specific privilege by more than one Role or Privilege assignment, you will be able to see this reflected in the Role Containment list.



Figure 6.37: Nexus User Privilege Trace

6.11 Network Configuration

By default, Nexus listens on port 8081. You can change this port, by changing the value in `$NEXUS_HOME/conf/nexus.properties`. This file is shown in [Contents of conf/nexus.properties](#). To change the port, stop Nexus, change the value of `applicationPort` in this file, and then restart Nexus. Once you do this, you should see a log statement in `$NEXUS_HOME/logs/wrapper.log` telling you that Nexus is listening on the altered port.

Contents of conf/nexus.properties

```
# Sonatype Nexus
# =====
```

```
# This is the most basic configuration of Nexus.

# Jetty section
application-port=8081
application-host=0.0.0.0
nexus-webapp=${bundleBasedir}/nexus
nexus-webapp-context-path=/nexus

# Nexus section
nexus-work=${bundleBasedir}/../sonatype-work/nexus
runtime=${bundleBasedir}/nexus/WEB-INF
```

6.12 Nexus Logging Configuration

You can configure the format and level of logging from within the Nexus interface. To do this, click on Log Configuration under the Administration menu in the left-hand navigation menu. Clicking on this link will display the panel shown in Figure 6.38.

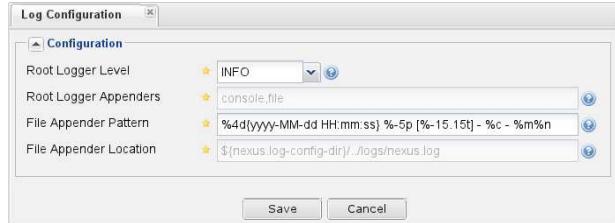


Figure 6.38: The Log Configuration Panel

From this panel you can configure the following logging configuration properties:

Root Logger Level

This controls how verbose the Nexus logging will be. If set to DEBUG, Nexus will be very verbose printing all log messages include debugging statements. If set to ERROR, Nexus will be far less verbose only printing out a log statement if Nexus encounters an error.

File Appender Pattern

This field controls the format of each log line. This field's format corresponds to the format ex-

pected by a Log4J PatternAppender. For more information about this format, refer to the [Javadoc for Log4J's PatternAppender](#).

The other configuration parameters: Root Logger Appenders and File Append Location, are not editable.

6.13 Nexus Plugins and REST Interfaces

As documented in Section [3.12](#) Nexus is built as a collection of plugins supported by a core architecture and additional plugins can be installed.

You can use the Nexus Plugin Console to list all installed Nexus plugins and browse REST services made available by installed Nexus Plugin. To open the Nexus Plugin Console, click on the Plugin Console link in the Administration section of the Nexus menu as shown in Figure [6.1](#).

Once you open the Plugin Console, you will see a list of plugins installed in your Nexus installation. Clicking on a plugin in this list will display information about the plugin including name, version, status, a description, SCM information about the plugin, and the URL of the plugin's project web site and links to the plugin documentation.

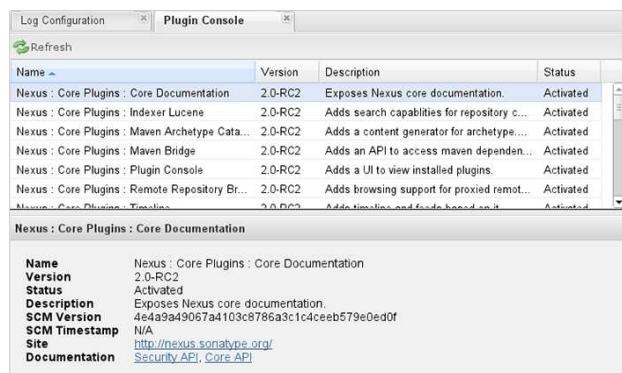


Figure 6.39: Plugin Console

An example for the plugin documentation is the Core API documentation linked off the Core Documentation plugin from Figure [6.39](#) and displayed in Figure [6.40](#)

The screenshot shows a web page titled "Sonatype Nexus OSS REST API". The left sidebar contains navigation links for "Home", "Introduction", "downloads", and "REST Endpoints". Under "REST Endpoints", there is a list of URLs. The main content area is titled "REST" and contains a detailed description of the REST API model, mentioning the support for the Representational State Transfer (REST) model and listing various resources accessible through the RESTful interface.

REST Endpoints

- [/all_repositories](#)
- [/artifact/maven](#)
- [/artifact/maven/content](#)
- [/artifact/maven/redirect](#)
- [/artifact/maven/resolve](#)
- [/attributes](#)
- [/attributes/{domain}/{target}/content](#)
- [/authentication/login](#)
- [/check_smtp_settings](#)

REST

This API supports a [Representational State Transfer \(REST\)](#) model for accessing a set of resources through a fixed set of operations. The following resources are accessible through the RESTful model:

- [/all_repositories](#)
- [/artifact/maven](#)
- [/artifact/maven/content](#)
- [/artifact/maven/redirect](#)
- [/artifact/maven/resolve](#)
- [/attributes](#)
- [/attributes/{domain}/{target}/content](#)
- [/authentication/login](#)
- [/check_smtp_settings](#)
- [/components/realm_types](#)
- [/components/repo_content_classes](#)
- [/components/repo_types](#)
- [/components/schedule_types](#)
- [/configs](#)
- [/configs/{configName}](#)

Figure 6.40: Documentation Website for the Core API

Chapter 7

Nexus Smart Proxy

7.1 Introduction

Typically an organization runs one single Nexus instance to proxy external artifacts as well as host internally produced artifacts. When a build is running against this Nexus instance it will look for any new artifacts in the proxied remote repositories. This adds additional network traffic that in many cases will just be a response from the remote server indicating that there are no changes.

This polling approach is fine for smaller deployment even though performance is impacted by this behaviour. Depending on your proxy timeout settings this polling won't actually result in updated artifacts as soon as they become available upstream either. If you try to support distributed instances, the only way you are going to achieve assurance that everything is up to date is by setting you expiration times to zero and constantly polling.

With the increased usage of Nexus in globally distributed teams or used by projects that span multiple organizations, each with their own Nexus instance hosting their own artifacts as well as proxying the other servers, this polling can result in significant traffic and a performance hit for all involved servers.

The Smart Proxy feature replaces this constant polling approach with a Publish-Subscribe based messaging approach between Nexus instances sharing a mutual trust. The result is a significantly improved performance due to nearly immediate availability of upstream artifact information directly in the downstream Nexus instances.

With Smart Proxy enabled the repositories involved will be able to keep their content in sync without sacrificing performance. This in turn allows collocating proxy servers with the involved development teams reducing build times and improving availability and performance of the used repository servers. The payoff grows with the number of involved Nexus servers in the network.

7.2 Enabling Smart Proxy Publishing

In order to enable the Smart Proxy feature on your Nexus instance, you need to navigate to the global Smart Proxy configuration screen. It is available in the left hand navigation in the Enterprise section. Selecting Smart Proxy will show you the configuration screen displayed in Figure 7.1.

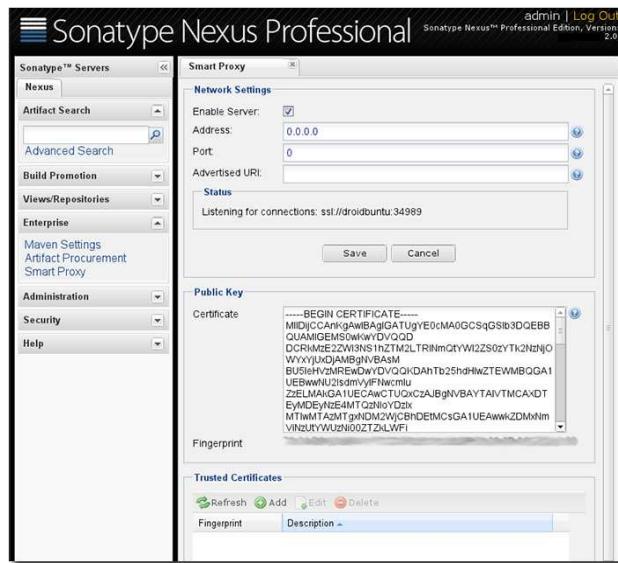


Figure 7.1: Global Configuration for Smart Proxy

The Network Settings section allows you to enable the Smart Proxy server with a check-box. This will need to be enabled on all servers that will publish events in the Smart Proxy network, while servers that will only act as subscribers can leave this option disabled.

In addition you can configure the address and port at which the publishing server will be available. The default address of 0.0.0.0 will cause the proxy to listen on all addresses. The default port number of 0 will

trigger usage of a random available port number for connection listening. If a random port is used, it will be chosen when the server (re)starts.

With the Advertised URI field it is possible to configure a specific address to be broad-casted by the proxy to the subscribing Smart Proxy clients enabling e.g. usage of a publicly available fully qualified host-name including the domain or also just usage of an externally reachable IP number.

Tip

It is important to enable the configured connection ports on any firewall between the servers to allow the direct socket connection between the servers and to avoid using random ports.

The Status field below the form will show the current status of the Smart Proxy including the full address and port.

The Public Key field displays the key identifying this server instance. It is automatically populated with the certificate associated with the public/private key pair that is generated when the server is first run.

Tip

The key is stored in sonatype-work/nexus/conf/keystore/private.ks and identifies this server. If you copy the sonatype work folder from one server to another as part of a migration or a move from testing to staging or production you will need to ensure that keys are not identical between multiple servers. To get a new key generated simply remove the key-store file and restart Nexus.

7.3 Establishing Trust

The servers publishing as well as subscribing to events identify themselves with their Public Key. This key has to be registered with the other servers in the Trusted Certificates section of the Smart Proxy configuration screen.

To configure two Nexus repository servers as trusted Smart Proxies, you copy the Public Key from the Certificate filed of the other server in the Trusted Certificates configuration section by adding a new certificate with a meaningful description as displayed in Figure 7.2 and Figure 7.3.

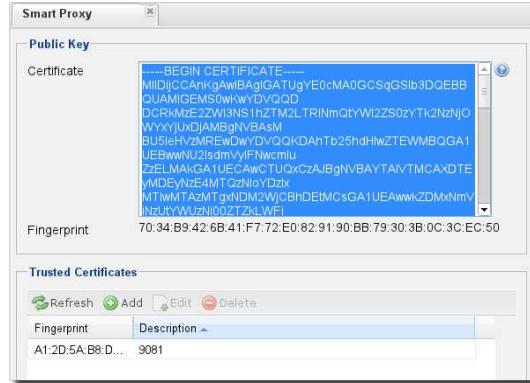


Figure 7.2: Copying a Certificate

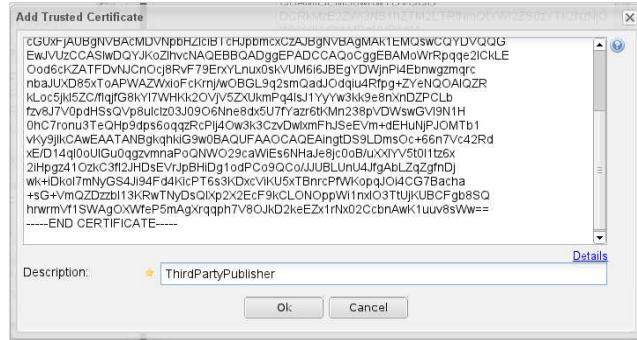


Figure 7.3: Adding a Trusted Certificate

All the key generation and certificates related to the trust management as handled by Nexus itself and no external configuration or usage of external keys is necessary or possible.

7.4 Repository Specific Smart Proxy Configuration

Once Smart Proxy has been configured and enabled as described above, you have to configure, which repositories' content should be synchronized between the servers. This is done in the Repositories ad-

ministration interface in a separate configuration tab titled Smart Proxy, which allows you to configure repository specific details as compared to server wide details described above.

On the publishing Nexus server you have to enable Smart Proxy on the desired hosted, virtual or proxy repositories in the repository configuration. This is accomplished by selecting the Publish Updates check-box in the Publish section of the Smart Proxy configuration for a specific repository as displayed in Figure 7.4 and pressing save.

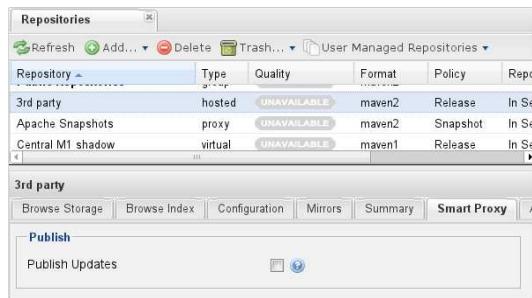


Figure 7.4: Smart Proxy Settings for a Hosted Repository

On the Nexus instance subscribing to the publishing server you have to create a new proxy repository to hold the synchronized artifacts. The Smart Proxy configuration for this repository displayed in Figure 7.5 allows you to activate the Receive Updates check-box in the Subscribe configuration section.



Figure 7.5: Smart Proxy Settings for a Proxy Repository

With a working trust established between the publishing and subscribing Nexus servers the Smart Proxy configuration of the proxy repository on the subscribing Nexus will display connection status as displayed in Figure 7.6.

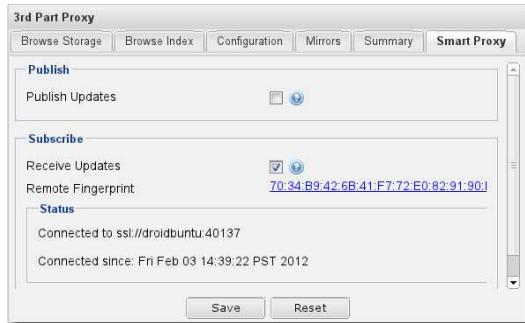


Figure 7.6: Subscription with Smart Proxy Connected

7.5 Smart Proxy Security and Messages

Smart Proxy messages are started with an initial handshake via HTTP. This handshake allows the two server to exchange their keys and confirm that they are configured with a valid trust relationship to communicate. After successful handshake, messages are sent in the middleware layer and can be configured to be sent via SSL encrypted messages.

The following events are broad-casted via Smart Proxy.

- a new artifact has been deployed
- an artifact has been deleted
- an artifact has been changed
- repository cache or a part of it has been cleared
- Smart Proxy publishing has been disabled

On the recipient side this will cause the changes to be applied mimicking what happened on the publisher. If Smart Proxy is disabled the subscription will be stopped.

7.6 Example Setup

In our example we are going to configure Smart Proxy for two Nexus servers, Server A and Server B. These servers provide proxying of external repositories like the Central Repository to the two development teams Team A and Team B. Both team deploy SNAPSHOT as well as release artifacts to their own Server A and Server B in the respective release and snapshot repositories. These snapshot and release artifacts are needed by the other team respectively. In addition both servers receive third party artifacts from another company wide Nexus instance for third party artifacts called Server T.

To configure Smart Proxy between these three servers you have to

- add the public key of Server A as trusted certificate to Server B and Server T
- add the public key of Server B as trusted certificate to Server A and Server T
- add the public key of Server T as trusted certificate to Server A and Server B
- set up the third party repository on Server T to publish updates
- set up a new third party proxy repository on Server A and Server B and configure it to receive updates from Server T
- set up a new snapshot and a new release proxy repository on Server A that proxies Server B
- set up a new snapshot and a new release proxy repository on Server B that proxies Server
- add all the new proxy repositories to the public group on Server A and Server B
- configure the snapshot and release hosted repositories on Server A and Server B to publish updates
- configure the new snapshot and new release proxy repositories on Server A and Server B to receive updates

With this setup the snapshot and release repositories of both servers will be available to both teams and always be synchronized and up to date. In addition both servers will have up to date content in their third party proxy repository.

Chapter 8

Nexus LDAP Integration

8.1 Introduction

Nexus Open Source has a Lightweight Directory Access Protocol (LDAP) Authentication realm which provides Nexus with the capability to authenticate users against an LDAP server. In addition to handling authentication, Nexus can be configured to map Nexus roles to LDAP user is a member of a group that matches the ID of a Nexus role, Nexus will grant that user the matching Nexus Role. In addition to this highly configurable user and group mapping capability, Nexus can augment LDAP group membership with Nexus-specific user-role mapping.

Nexus Professional offers LDAP support features for enterprise LDAP deployments including the ability to cache authentication information, support for multiple LDAP servers and backup mirrors, the ability to test user logins, support for common user/group mapping templates, and the ability to support more than one schema across multiple servers.

8.2 Enabling the LDAP Authentication Realm

Authentication realm, you will need to add the Nexus LDAP Authentication Realm to the Selected Realms in the Security section of the Server configuration panel. To load the Server configuration panel, click on the Server link under Administration in the Nexus menu. Once you have the Server configuration panel

loaded, select OSS LDAP Authentication Realm in the Available Realms list under the Security section and click the Add button (or Left Arrow) as shown in Figure 8.1.

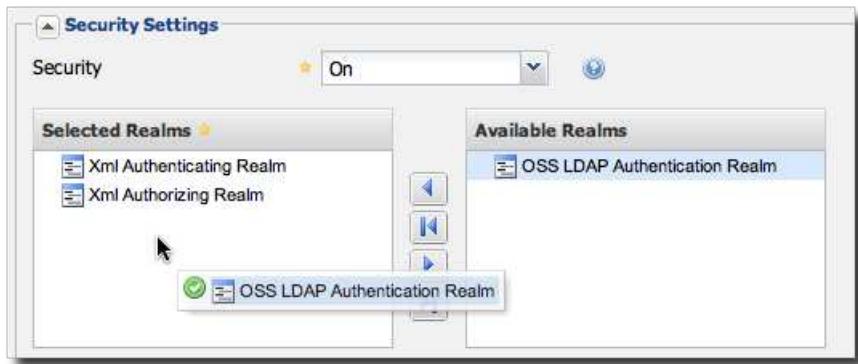


Figure 8.1: Adding the LDAP Authentication Realm to Available Realms

Next, click on the Save button in the Server configuration panel. To make sure that your Nexus instance attempts to authenticate against LDAP first, click on the OSS LDAP Authentication Realm in the Selected Realms list and drag it to the top of the list as shown in Figure 8.2. This will ensure that Nexus will consult the LDAP Server before it consults the other Authentication Realms.

Note

Nexus will continue to fall-back to the Xml Authenticating Realm if it cannot authenticate a user against the LDAP server. If your LDAP server is unavailable, and you need to reconfigure the LDAP connection, you can always login as the default admin

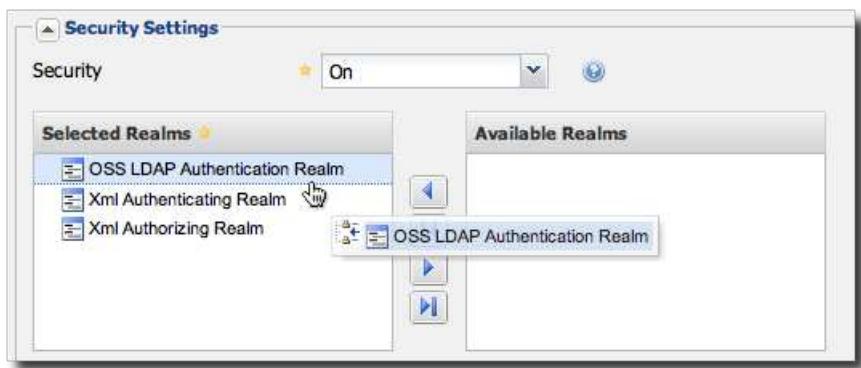


Figure 8.2: Move the LDAP Authentication Realm to the Top of Selected

8.3 Configuring Nexus LDAP Integration

To configure LDAP integration, click on the LDAP Configuration link in the Nexus menu as shown in Figure 8.3. LDAP Configuration is under Security in the Nexus menu.



Figure 8.3: LDAP Configuration Option in the Nexus Menu

Clicking on the LDAP Configuration link will load the LDAP Configuration panel as shown in [?informalfigure] and [?informalfigure]. The following sections outline the configuration options available in the LDAP Configuration Panel.

8.4 Connection and Authentication

The following figure shows Nexus configured to connect to an LDAP server running on localhost port 10389 using the search base of "ou=system". On a more standard installation, you would likely not want to use Simple Authentication as it sends the password in clear text over the network, and you would also use a search base which corresponds to your organization's top-level domain components such as "dc=sonatype,dc=com".

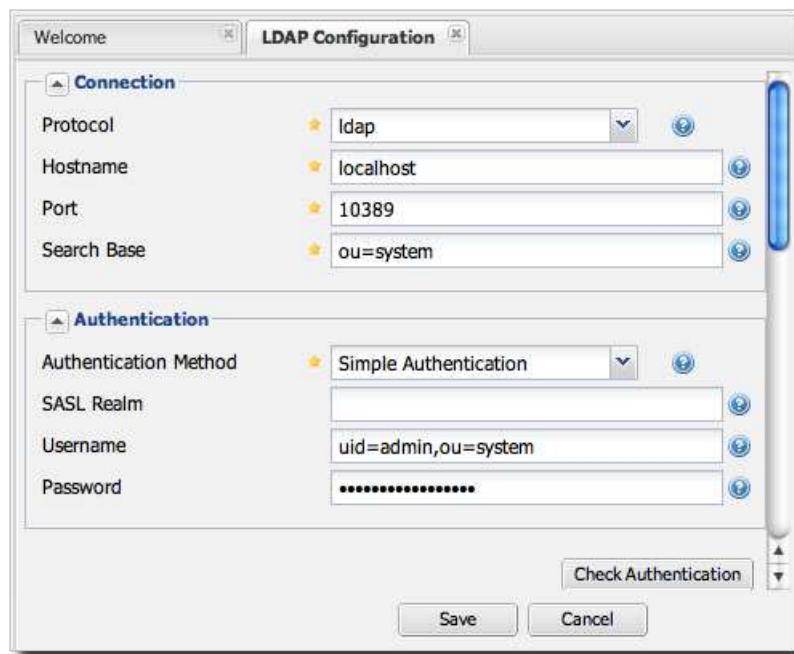


Table 8.1 and Table 8.2 contain detailed descriptions of the configuration fields in both the Connection and Authentication sections of the LDAP Configuration panel.

Table 8.1: Connection Configuration for LDAP Integration

Field Name	Description
Protocol	Valid values in this drop-down are ldap and ldaps which correspond to the Lightweight Directory Access Protocol and the Lightweight Directory Access Protocol over SSL
Hostname	The hostname or IP address of the LDAP

Table 8.1: (continued)

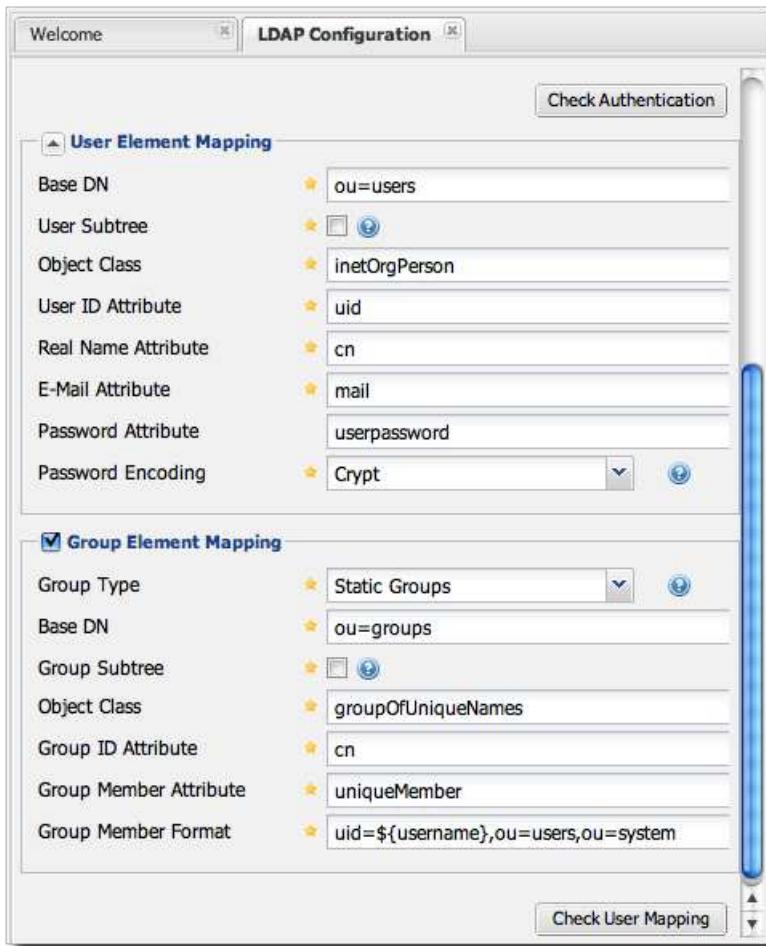
Field Name	Description
Port	The port on which the LDAP server is listening. Port 389 is the default port for the ldap protocol, and port 636 is the default port for the ldaps
Search Base	The search base is the Distinguished Name (DN) to be appended to the LDAP. The search base usually corresponds to the domain name of an organization. For example, the search base on the Sonatype LDAP server is "dc=sonatype,dc=com"

Table 8.2: Authentication Configuration for LDAP Integration

Field Name	Description
Authentication Method	<p>Nexus provides four distinct authentication methods to be used when connecting to the LDAP Server:</p> <p>Simple Authentication:: Simple authentication is not recommended for production deployments not using the secure ldaps protocol as it sends a clear-text password over the network.</p> <p>Anonymous Authentication:: Used when Nexus only needs read-only access to non-protected entries and attributes when binding to the LDAP</p> <p>Digest-MD5:: This is an improvement on the CRAM-MD5 authentication method. For more information, see http://www.ietf.org/rfc/rfc2831.txt</p> <p>CRAM-MD5:: The Challenge-Response Authentication Method (CRAM) 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 which the server compares to an expected value. For more information, see RFC 2195</p> <p>For a full discussion of LDAP authentication approaches, see http://www.ietf.org/rfc/rfc2829.txt and http://www.ietf.org/rfc/rfc2251.txt</p>
SASL Realm	The Simple Authentication and Security Layer (SASL) Realm to connect with. The SASL Realm is only available if the authentication method is Digest-MD5 or CRAM-MD5.
Username	Username of an LDAP User to connect (or bind) with. This is a Distinguished Name of a user who has read access to all users and groups
Password	Password for an Administrative LDAP User

8.5 User and Group Mapping

The LDAP Configuration panel also contains sections to manage User Element Mapping and Group Element Mapping as shown in [?informalfigure].



The fields for both the User Element Mapping and Group Element Mapping sections are described in detail in Table 8.3 and Table 8.4.

Table 8.3: User Element Mapping Configuration for LDAP Integration

Field Name	Description
Base DN	Corresponds to the Base DN containing user entries. This DN is going to be relative to the Search Base which was specified in [?informalfigure]. For example, if your users are all contained in "ou=users,dc=sonatype,dc=com" and you specified a Search Base of "dc=sonatype,dc=com" you would use a value of "ou=users"
User Subtree	True if there is a tree below the Base DN which can contain user entries. False if all users are contained within the specified Base DN. For example, if all users are in "ou=users,dc=sonatype,dc=com" this field should be false. If users can appear in organizational units within organizational units such as "ou=development,ou=users,dc=sonatype,dc=com" this field should be true.
Object Class	This value defaults to inetOrgPerson which is a standard object class defined in RFC 2798 . inetOrgPerson contains standard fields such as mail, uid. Other possible values are posixAccount or a custom class.
User ID Attribute	This is the attribute of the Object class which supplies the User ID. Nexus will use this attribute as the Nexus User ID.
Real Name Attribute	This is the attribute of the Object class which supplies the real name of the user. Nexus will use this attribute when it needs to display the real name of a user.
E-Mail Attribute	This is the attribute of the Object class which supplies the email address of the user. Nexus will use this attribute when it needs to send an email to a user.
Password Attribute	This is the attribute of the Object class which supplies the password of the user. Nexus will use this attribute when it is authenticating a user against an LDAP server.
Password Encoding	Defines the preferred password encoding mechanism to be used when sending password data to the LDAP server.

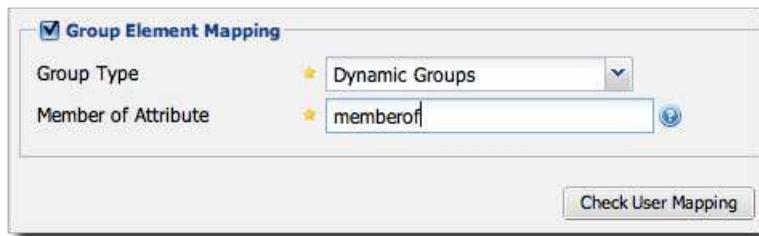
Table 8.4: Group Element Mapping Configuration for LDAP Integration

Field Name	Description
Group Type	Groups are generally one of two types in LDAP systems - static or dynamic. A static group contains a list of users. A dynamic group is where the user contains a list of groups the user belongs to. In LDAP a static group would be captured in an entry with an Object class groupOfUniqueNames which contains one or more uniqueMember attributes. In a dynamic group configuration, each user entry in LDAP contains an attribute which lists group membership.
Base DN	This field is similar to the Base DN field described in Table 8.3. This field is visible if Static Groups is selected. 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 in Table 8.3. If all groups are defined under the entry defined in Base DN, this field should be false, if a group can be defined in a tree of organizational units under the Base DN, this field should be true. This field is visible if Static Groups is selected.
Object Class	This value defaults to groupOfUniqueNames which is a standard object class defined in RFC 4519 groupOfUniqueNames is simply a collection of references to unique entries in an LDAP directory and can be used to associate user entries with a group. Other possible values are posixGroup or a custom class. This field is visible if Static Groups is selected.
Group ID Attribute	Specifies the attribute of the Object class which specifies the Group ID. If the value of this field corresponds to the ID of a Nexus Role, members of this group will have the corresponding Nexus privileges. Defaults to "cn". This field is visible if Static Groups is selected.
Group Member Attribute	Specifies the attribute of the Object class which specifies a member of a group. A groupOfUniqueNames has multiple uniqueMember attributes for each member of a group. Defaults to "uniqueMember". This field is visible if Static Groups is selected.
Group Member Format	This field captures the format of the Group Member Attribute and it is used by Nexus to extract a username from this attribute. For example, if the Group Member Attribute has the format "uid=brian,ou=users,dc=sonatype,dc=com", then the Group Member Format would be "uid=\$username,ou=users,dc=sonatype,dc=com". If the Group Member Attribute had the format "brian", then the Group Member Format would be "\$username". This field is visible if Static Groups is selected.

Table 8.4: (continued)

Field Name	Description
Member of Attribute	When Dynamic Groups is selected, Nexus will inspect an attribute of the user entry to get a list of groups that the user is a member of. In this configuration, a user entry would have an attribute such as memberOf which would contain the name of a group. This field is visible if Dynamic Groups is selected.

If your installation does not use Static Groups, you can configure Nexus LDAP Integration to refer to an attribute on the User entry to derive group membership. To do this, select Dynamic Groups in the Group Type field in Group Element Mapping. Selecting Dynamic Groups will show a single field named Member of Attribute as shown in [?informalfigure].



8.6 Mapping Users and Groups with Active Directory

When mapping users and groups to an Active Directory installation, try the common configuration values listed in Table 8.6 and Table 8.7.

Table 8.5: Connection and Authentication Configuration for Active Directory

Configuration Element	Configuration Value
Protocol	ldap
Hostname	Hostname of Active Directory Server
Port	389 (or port of AD server)

Table 8.5: (continued)

Configuration Element	Configuration Value
Search Base	DC=yourcompany,DC=com (customize for your organization)
Authentication	Simple Authentication
Username	CN=Administrator,CN=Users,DC=yourcompany,DC=com

Table 8.6: User Element Mapping Configuration for Active Directory

Configuration Element	Configuration Value
Base DN	cn=users
User Subtree	false
Object Class	user
User ID Attribute	sAMAccountName
Real Name Attribute	cn
E-Mail Attribute	mail
Password Attribute	(Not Used)
Password Encoding	Crypt

Table 8.7: Group Element Mapping Configuration for Active Directory

Configuration Element	Configuration Value
Group Type	Dynamic Groups
Member Of Attribute	memberOf

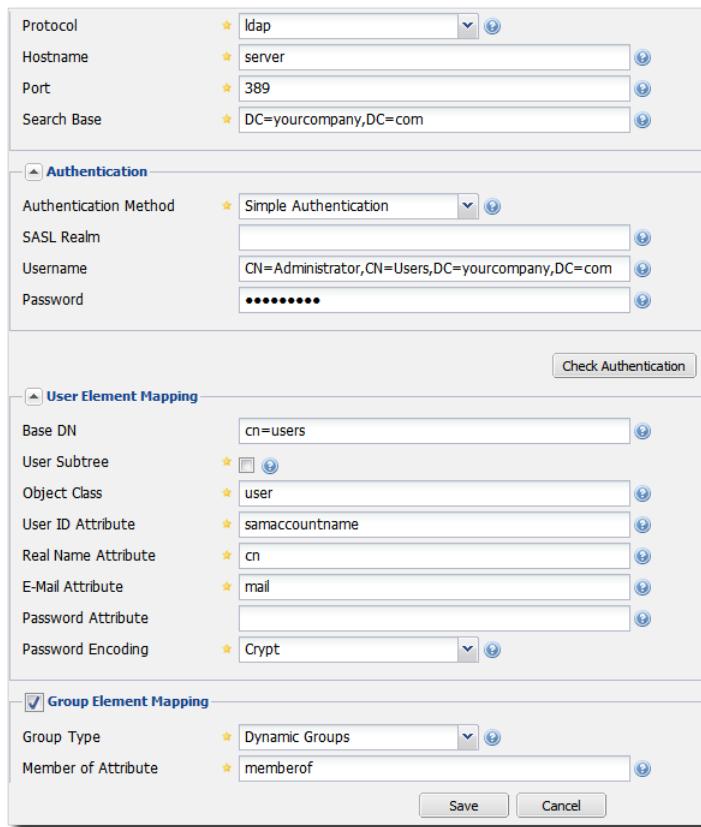


Figure 8.4: Configuring Nexus LDAP for Active Directory



Warning

You should connect to the AD through port 3268 if you have a multi-domain, distributed Active Directory forest. Connecting directly to port 389 might lead to errors.

8.7 Mapping Users and Groups with `posixAccount`

When mapping users and groups to LDAP entries of type `posixAccount`, try the common configuration values listed in Table 8.8 and Table 8.9.

Table 8.8: User Element Mapping Configuration for posixAccount

Configuration Element	Configuration Value
Base DN	(Not Standard)
User Subtree	false
Object Class	posixAccount
User ID Attribute	sAMAccountName
Real Name Attribute	uid
E-Mail Attribute	mail
Password Attribute	(Not Used)
Password Encoding	(Not Used)

Table 8.9: Group Element Mapping Configuration for posixGroup

Configuration Element	Configuration Value
Group Type	Static Groups
Base DN	(Not Standard)
Group Subtree	false
Object Class	posixGroup
Group ID Attribute	cn
Group Member Attribute	memberUid
Group Member Format	

8.8 Mapping Roles to LDAP Users

Once User and Group Mapping has been configured, you can start verifying how LDAP users and groups are mapped to Nexus Roles. If a user is a member of an LDAP group that has a Group ID corresponding to the ID of a Nexus Role, that user is granted the appropriate permissions in Nexus. For example, if the LDAP user entry in "uid=brian,ou=users,dc=sonatype,dc=com" is a member of a groupOfUniqueNames attribute value of admin, when this user logs into Nexus, it will be granted the Nexus Administrator Role if Group Element Mapping is configured properly. To verify the User Element Mapping and Group Element Mapping, click on Check User Mapping in the LDAP Configuration panel directly below the Group Element Mapping section, Figure 8.5 shows the results of this check.

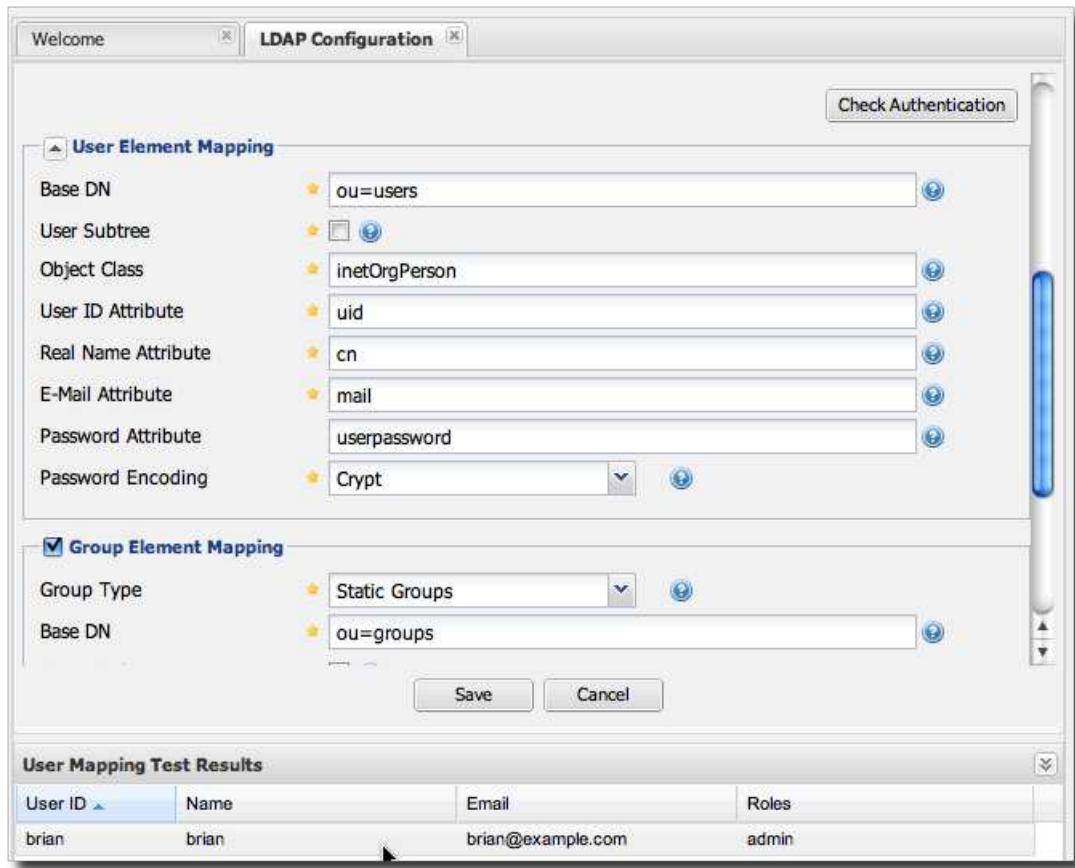


Figure 8.5: Checking the User and Group Mapping in LDAP Configuration

In Figure 8.5, Nexus LDAP Integration locates a user with a User ID of "brian" who is a member of the "admin" group. When brian logs in, he will have all of the rights that the admin Nexus Role has.

8.9 Mapping Nexus Roles for External Users

If you are unable to map all of the Nexus roles to LDAP groups, you can always augment the Role information by adding a specific user-role mapping for an external LDAP user in Nexus. In other words, if you need to make sure that a specific user in LDAP gets a specific Nexus role and you don't want to model

this as a group membership, you can add a role mapping for an external user in Nexus. Nexus will keep track of this association independent of your LDAP server. Nexus continues to delegate authentication to the LDAP server for this user, Nexus will continue to map the user to Nexus roles based on the group element mapping you have configured, but Nexus will also add any roles specified in the User panel. You are augmenting the role information that Nexus gathers from the group element mapping.

Once the User and Group Mapping has been configured, click on the Users link under Security in the Nexus menu. The Users tab is going to contain all of the "configured" users for this Nexus instance as shown in Figure 8.6. A configured user is a user in a Nexus-managed Realm or an External User which has an explicit mapping to a Nexus role. In Figure 8.6, you can see the three default users in the Nexus-managed default realm plus the brian user from LDAP. The brian user appears because this user has been mapped to a Nexus role.

The screenshot shows the Nexus 'Users' interface. At the top, there's a toolbar with 'Welcome', 'Users' (selected), 'Refresh', 'Add...', 'Delete', 'All Configured Users' (set to 'All'), and a search bar. Below the toolbar is a table titled 'All Configured Users' with columns: User ID, Realm, Name, Email, and Roles. The table contains four rows:

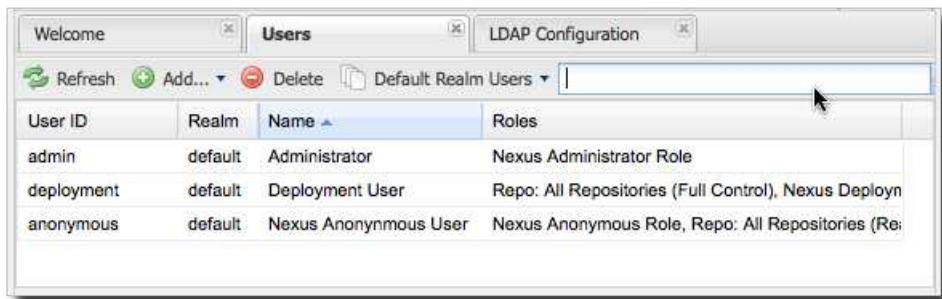
User ID	Realm	Name	Email	Roles
admin	default	Administrator	changeme@yourcompany.com	Nexus Administrator Role
brian	LDAP	Brian Fox		Nexus Deployment Role
deployment	default	Deployment User	changeme1@yourcompany.com	Repo: All Repositories (f)

Below the table, a 'Deployment User' form is displayed. It includes fields for User ID (deployment), Name (Deployment User), Email (changeme1@yourcompany.com), and Status (Active). There are two panels at the bottom: 'Selected Roles' containing 'Nexus Deployment Role' and 'Available Roles' containing 'Nexus Administrator Role'. Buttons for 'Save' and 'Reset' are at the bottom right.

Figure 8.6: Viewing All Configured Users

The list of users in Figure 8.6 is a combination of all of the users in the Nexus default realm and all of the External Users with role mappings. To explore these two sets of users, click on the All Configured Users drop-down and choose "Default Realm Users". Once you select this, click in the search field and press Enter. Searching with a blank string in the Users panel will return all of the users of the selected type. In

Figure 8.7 you see a dialog containing all three default users from the Nexus default realm.



The screenshot shows a window titled 'Users' with a sub-tab 'Default Realm Users'. The table lists three users:

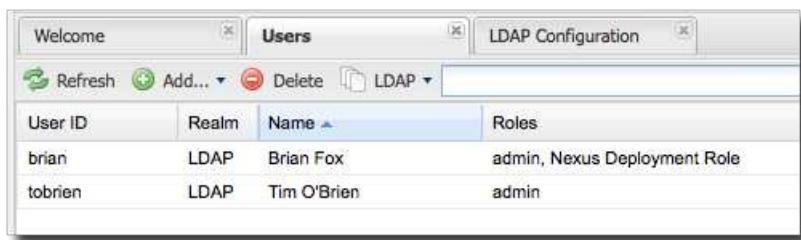
User ID	Realm	Name	Roles
admin	default	Administrator	Nexus Administrator Role
deployment	default	Deployment User	Repo: All Repositories (Full Control), Nexus Deploy
anonymous	default	Nexus Anonymous User	Nexus Anonymous Role, Repo: All Repositories (Re

Figure 8.7: All Default Realm Users

If you wanted to see a list of all LDAP users, select "LDAP" from the "All Configured Users" drop-down shown in Figure 8.6 and click on the search button (magnifying glass) with an empty search field. Clicking search with an empty search field will return all of the LDAP users as shown in Figure 8.8.

Note

Note that the user "tobrien" does not show up in the "All Configured Users" list. This is by design. Nexus is only going to show you information about users with external role mappings. If an organization has an LDAP directory with thousands of developers, Nexus doesn't need to retain any configuration information for users that don't have custom Nexus role mappings.



The screenshot shows a window titled 'Users' with a sub-tab 'LDAP'. The table lists two users:

User ID	Realm	Name	Roles
brian	LDAP	Brian Fox	admin, Nexus Deployment Role
tobrien	LDAP	Tim O'Brien	admin

Figure 8.8: All LDAP Users

To add a mapping for an external LDAP user, you would click on the "All Configured Users" drop-down and select LDAP. Once you've selected LDAP, type in the user ID you are searching for and click the

search button (magnifying glass icon to right of the search field). In Figure 8.9, a search for "brian" yields one user from the LDAP server.

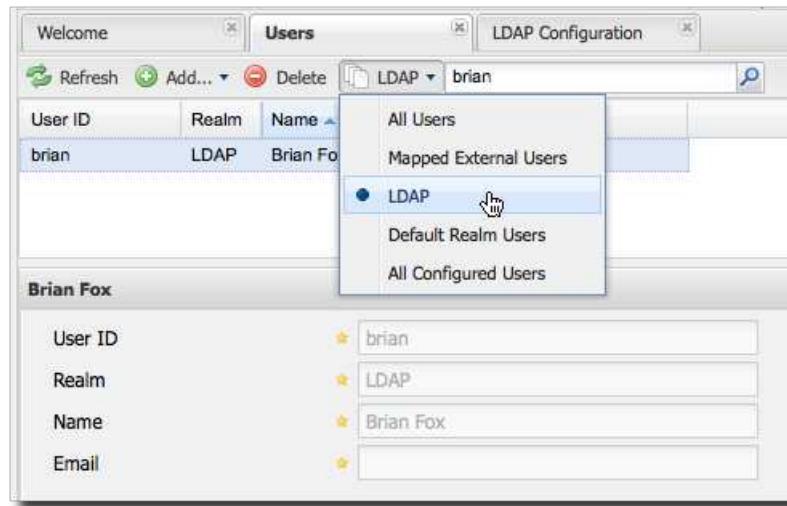


Figure 8.9: Search LDAP Users

To add a Nexus role mapping for the external user "brian" shown in Figure 8.9, click on the user in the results table and drag a role from Available Roles to Selected Roles as shown in Figure 8.10. In this case, the user "brian" is mapped to the Administrative group by virtue of his membership in an "admin" group in the LDAP server. In this use case, a Nexus administrator would like to grant Brian the Deployment Role without having to create a LDAP group for this role and modifying his group memberships in LDAP

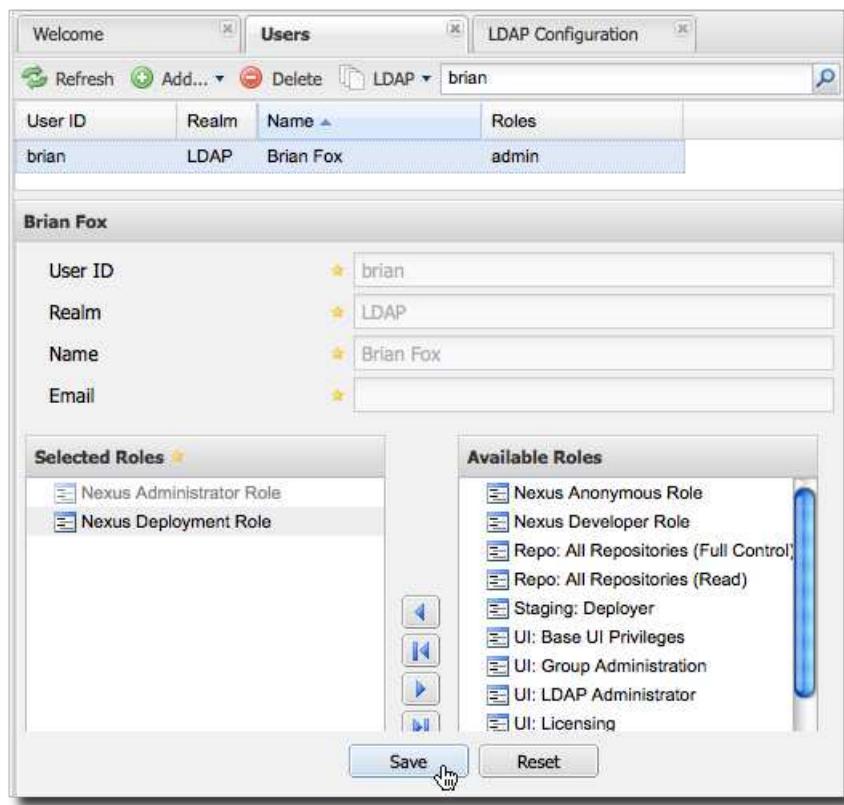


Figure 8.10: Mapping the Deployment Role to an External User

The end result of this operation is to augment the Group-Role mapping that is provided by the LDAP integration. You can use LDAP groups to manage coarse-grained permissions to grant people administrative privileges and developer roles, and if you need to perform more targeted privilege assignments in Nexus you can Map LDAP users to Nexus roles with the techniques shown in this section.

8.10 Mapping External Roles to Nexus Roles

Nexus makes it very straightforward to map an external role to an internal Nexus role. This is something you would do, if you want to grant every member of an externally managed group (such as an LDAP group) a certain privilege in Nexus. For example, assume that you have a group in LDAP named "svn"

and you want to make sure that everyone in the "svn" group has Nexus Administrative privileges. To do this, you would click on the Add.. drop-down in the Role panel as shown in Figure 8.11. This drop-down can be found in the Role management panel which is opened by clicking on Roles in the Security menu.

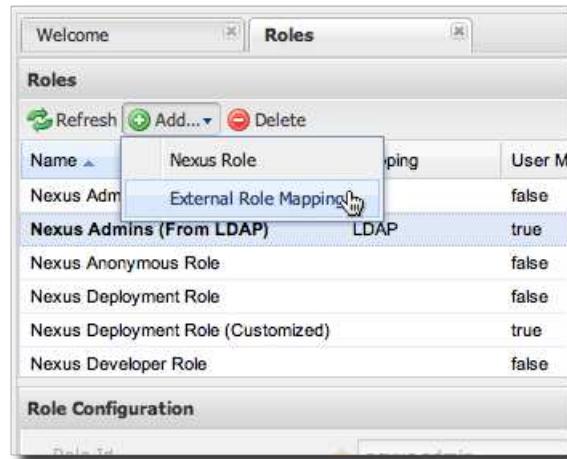


Figure 8.11: Selecting External Role Mapping in the Role Management Panel

Selecting External Role Mapping under Add... will show you a dialog which contains a drop-down of External Realms. Selecting an external realm such as LDAP will then bring up a list of roles managed by that external realm. The dialog shown in Figure 8.12 shows the external realm LDAP selected and the role "svn" being selected to map to a Nexus role.

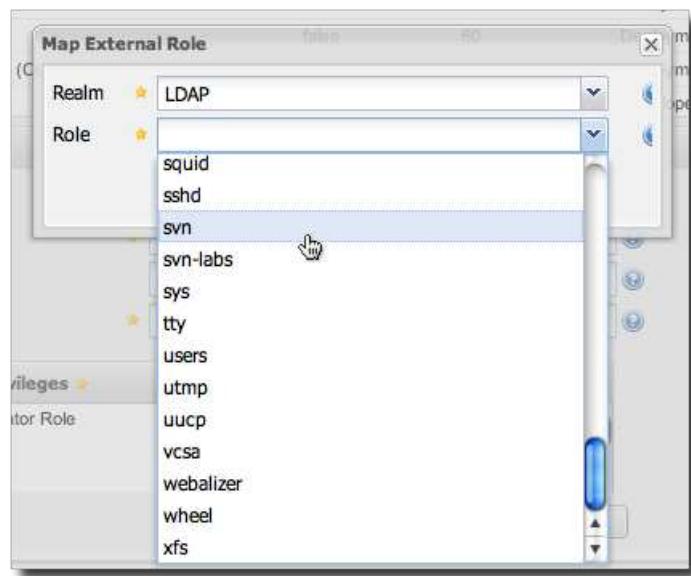


Figure 8.12: Selecting an Externally Managed Role to Map to a Nexus Role

Once the external role has been selected, Nexus will create a corresponding Nexus Role. You can then assign other Roles to this new externally mapped Role. Figure 8.13 shows that the SVN role from LDAP is being assigned the Nexus Administrator Role. This means that any user that is authenticated against the external LDAP Realm who is a member of the svn LDAP group will be assigned a Nexus role that maps to the Nexus Administrator Role.

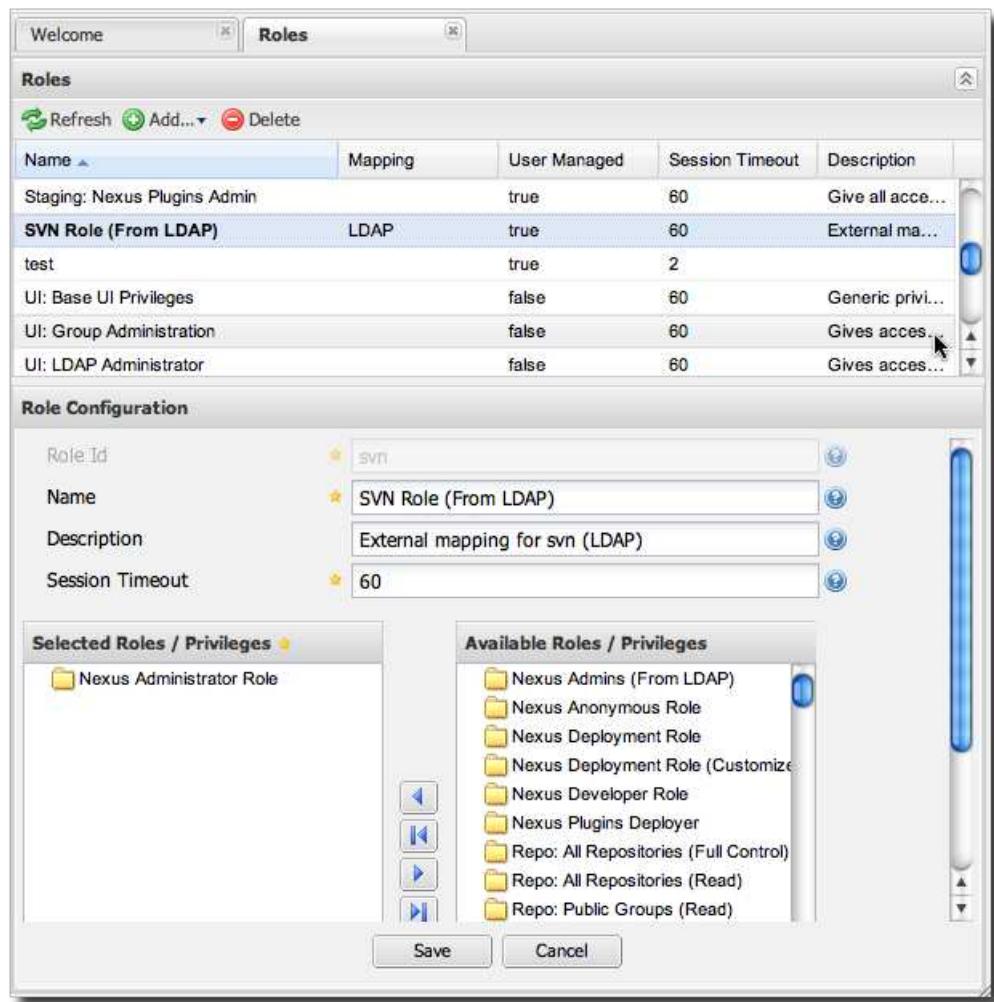


Figure 8.13: Mapping an External Role to a Nexus Role

8.11 Enterprise LDAP Support

The following sections outline Enterprise LDAP features which are available in Nexus Professional. Enterprise LDAP Fail-over Support When an LDAP server fails, the applications authenticating against it can also become unavailable. Because a central LDAP server is such a critical resource, many large

software enterprises will install a series of primary and secondary LDAP servers to make sure that the organization can continue to operate in the case of an unforeseen failure. Nexus Professional's Enterprise LDAP plugin now provides you with the ability to define multiple LDAP servers for authentication. To configure multiple LDAP servers, click on Enterprise LDAP under Security in the Nexus application menu. You should see the Enterprise LDAP panel shown in the following figure.

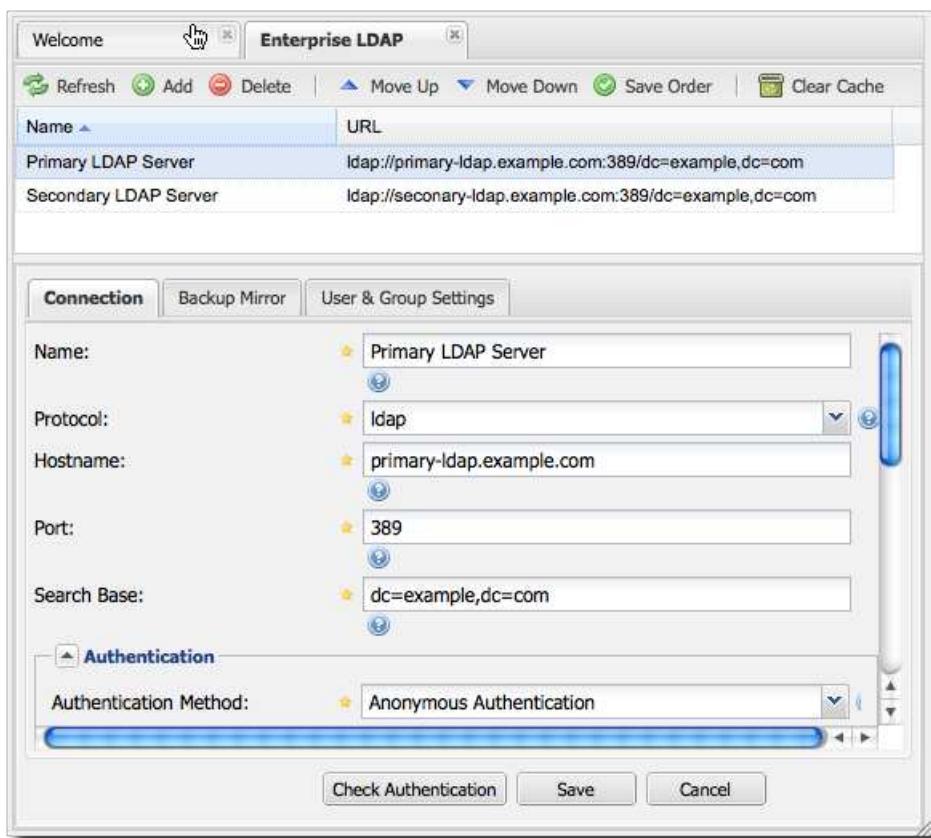


Figure 8.14: Defining Multiple LDAP Servers in Nexus Professional

You can use the Backup Mirror setting for an LDAP repository. This backup mirror is another LDAP server which will be consulted if the original LDAP server cannot be reached. Nexus Professional assumes that the backup mirror is a carbon copy of the original LDAP server, and it will use the same user and group mapping configuration as the original LDAP server. Instead of using the backup mirror settings, you could also define multiple LDAP backup mirrors in the list of configured LDAP servers shown in the previous figure. When you configure more than one LDAP server, Nexus Professional will consult the servers in the order they are listed in this panel. If Nexus can't authenticate against the first LDAP server,

Nexus Professional will move on to the next LDAP server until it either reaches the end of the list or finds an LDAP server to authenticate against.

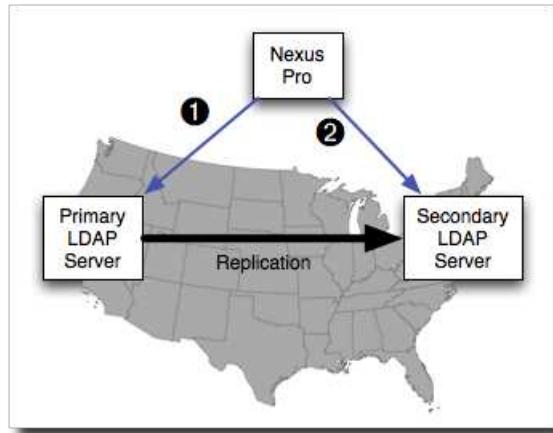


Figure 8.15: Use Multiple LDAP Servers in a Fail-over Scenario

The feature just described is one way to increase the reliability of your Nexus instance. In the previous case, both servers would have the same user and group information. The secondary would be a mirror of the primary. But, what if you wanted to connect to two LDAP servers that contained different data? Nexus Professional also provides...

8.11.1 Support for Multiple Servers and LDAP Schemas

The same ability to list more than one LDAP server also allows you to support multiple LDAP servers which may or may not contain the same user authentication information. Assume that you had an LDAP server for the larger organization which contained all of the user information across all of the departments. Now assume that your own department maintains a separate LDAP server which you use to supplement this larger LDAP installation. Maybe your department needs to create new users that are not a part of the larger organization, or maybe you have to support the integration of two separate LDAP servers that use different schema on each server.

A third possibility is that you need to support authentication against different schema within the same LDAP server. This is a common scenario for companies which have merged and whose infrastructures has not yet been merged. To support multiple servers with different user/group mappings or to support a single server with multiple user/group mappings, you can configure these servers in the Enterprise LDAP

panel shown above. Nexus will iterate through each LDAP server until it can successfully authenticate a user against an LDAP server.

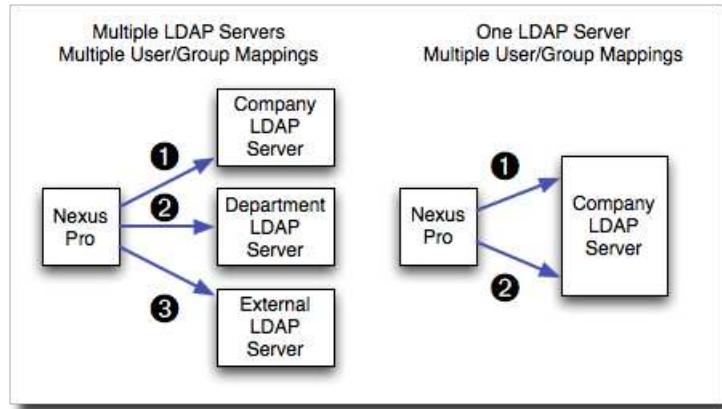


Figure 8.16: Supporting Multiple LDAP Schemas with Nexus Professional

8.11.2 Enterprise LDAP Performance Caching and Timeout

If you are constantly authenticating against a large LDAP server, you may start to notice a significant performance degradation. With Nexus Professional you can cache authentication information from LDAP. To configure caching, create a new server in the Enterprise LDAP panel, and scroll to the bottom of the Connect tab. You should see the following input field which contains the number of seconds to cache the results of LDAP queries.

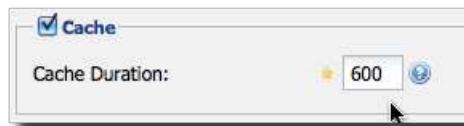


Figure 8.17: Setting the LDAP Query Cache Duration (in Seconds)

You will also see options to alter the connection timeout and retry interval for an LDAP server. If you are configuring a number of different LDAP servers with different user and group mappings, you will want to make sure that you've configured low timeouts for LDAP servers at the beginning of your Enterprise

LDAP server list. If you do this properly, it will take Nexus next to no time to iterate through the list of configured LDAP servers.



Figure 8.18: Setting the LDAP Connection Timeout (in Seconds)

We improved the overall caching in this release. The cache duration is configurable and applies to authentication and authorization, which translates into pure speed! Once you've configured LDAP caching in Nexus Professional, authentication and other operations that involve permissions and credentials once retrieved from an external server will run in no time.

8.11.3 User and Group Templates

If you are configuring your Nexus Professional instance to connect to an LDAP server there is a very good chance that your server follows one of several, well-established standards. Nexus Professional's LDAP server configuration includes these widely used user and group mapping templates which great simplify the setup and configuration of a new LDAP server. To configure user and group mapping using a template, select a LDAP server from the Enterprise LDAP panel, and choose the User and Group Settings. You will see a User & Group Templates section as shown in the following figure.

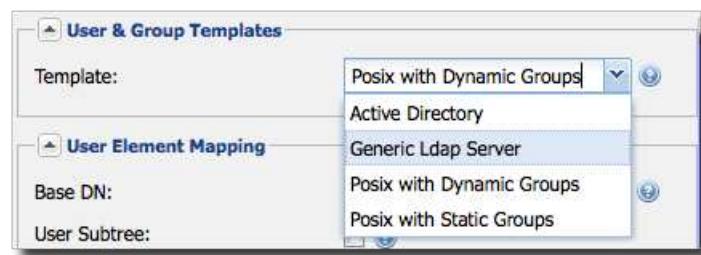


Figure 8.19: Using User & Group Mapping Templates

8.11.4 Testing a User Login

Nexus Professional provides you with the ability to test a user login directly. To test a user login, go to the User and Group Settings tab for a server listed in the Enterprise LDAP panel. Scroll to the bottom of the form, and you should see a button named "Check Login".

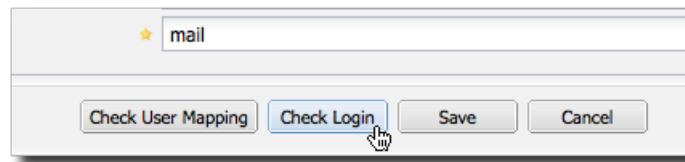


Figure 8.20: Testing a User Login

If you click on Check Login, you will then be presented with the login credentials dialog shown below. You can use this dialog to login as an LDAP user and test the user and group mapping configuration for a particular server. This feature allows you to test user and group mapping configuration directly. This feature allows you to quickly diagnose and address difficult authentication and access control issues via the administrative interface.

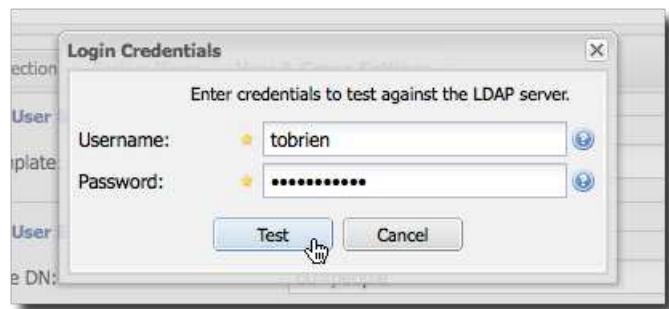


Figure 8.21: Supply a User's Login Credentials

Chapter 9

Nexus Procurement Suite

9.1 Introduction

Nexus Procurement Suite provides an organization with control over what artifacts are allowed into a repository from an external, proxied repository such as the Maven Central repository. Such control can be a prerequisite for organizations unwilling or unable to trust the entire contents of an external public repository. If an organization is developing mission critical code, they will likely want to subject every third party dependency to intense scrutiny and testing before making an artifact available to build a release or support a team of developers. In most Enterprise development environments, a developer can't just decide to add in a new dependency to Hibernate or to the Spring Framework on a whim; the decision to add dependencies to third-party libraries will need to be funnelled through an oversight process that relies on an architect or an administrator to promote artifacts to a certified release repository.

Another, more common experience is an organization which needs to proxy something like the Maven Central repository but wants to limit access to specific versions of artifacts or prevent dependencies on everything contained under a specific group. Some organizations are more amenable to trusting the contents of a remote, proxied repository like Central, but they also need the ability to block certain dependencies. Maybe you work on a team that needs to limit access to dependencies with a certain license, or maybe you just want to make sure no one uses a problematic version of Hibernate with a known bug? The procurement suite is the tool that provides for both coarse and fine-grained control of the artifacts that can appear in a repository.

9.2 The Stages of Procurement

A procured repository is a Hosted Repository which procures artifacts from a Proxy Repository. For example, one could create a hosted repository named "Secured Maven Central" and then configure this hosted repository to procure artifacts from the "Maven Central" repository. Once the hosted repository has been created and the source of procurement has been configured, the repository will obtain artifacts from the proxy repository as long as procurement is activated. If you start procurement for a Hosted repository, the hosted repository will fetch artifacts from the Proxy repository specified in the procurement settings. If you stop procurement for a Hosted Repository, no additional artifacts will be retrieved from the Proxy repository specified in the procurement settings.

The ability to enable or disable procurement for a Hosted repository comes in very handy when you want to "certify" a Hosted repository as containing all of the artifacts (no more and no less) required for a production build. You can start procurement, run a build which triggers artifact procurement, and then stop procurement knowing that the procured repository now contains all of the artifacts required for building a specific project. Stopping procurement assures you that the contents of the repository will not change if the third-party, external proxied repository does. This is an extra level of assurance that your release artifacts depend on a set of artifacts under your complete control.

9.3 Two Approaches to Procurement

There are two main use cases for the Procurement Suite. In the first use case, the Procured Release Repository, the Procurement features of Nexus Professional are used to create a procured release repository to make sure that the organization has full control over the artifacts that are making it into a production release. The other use case, the Procured Development Repository, is for organizations that need more up-front control over which artifacts are allowed during the development of a project. The following sections describe these two uses cases in more detail.

9.3.1 Procured Release Repository

The Procurement Suite can be used in two different ways. In the "Procured Release" mode, developers work with a proxied 3rd party repository exactly as they would without the Procurement Suite. When a developer needs to add a dependency on a new artifact, Nexus will retrieve the artifact from the third-party repository (like Central or Apache Snapshots) and this artifact will be served to Maven via a proxied Nexus repository. When a QA or Release engineer needs to build a release or staging artifact, the Release or QA build would be configured to execute against a procured repository. A procured repository is one

which only serves the artifacts which have been explicitly approved using the Procurement Suite.

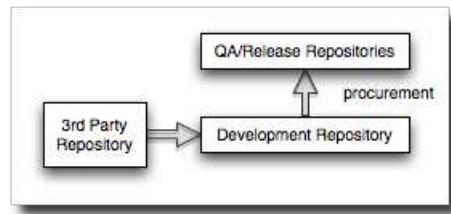


Figure 9.1: Procurement to a Certified Release Repository

In this model, developers can add as many third-party dependencies as they want, and it is the responsibility of the QA and Release engineers to approve (or procure) artifacts from the Development Repository to the QA/Release Repository. Developers can move forward, adding dependencies freely from a third-party, proxied repository, but once it is time to populate a Release repository, a Nexus administrator can audit the required artifacts, create a hosted repository, turn on procurement, populate the repository, and then deactivate procurement. This has the effect of "locking down" the artifacts that are involved in a production release.

9.3.2 Procured Development Repository

There are some development environments which require even more control over which artifacts can be used and referenced by developers. In these situations, it might make sense to only allow developers to work with a procured repository. In this mode, a developer must ask a Nexus administrator for permission to add a dependency on a particular third-party artifact. A Procurement Manager would then have to approve the artifact, or group of artifacts so that they would be made available to the developers. This is the "ask-first" model for organizations that want to control which artifacts make it into the development cycle.

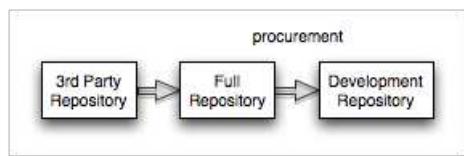


Figure 9.2: Procurement to a Certified Development Repository

This is a model common in industries that have strict oversight requirements. More often than not, banks, hospitals, and government agencies have fairly strict regulations on the software that can be used by large development teams. With the Procured Development Repository approach, an architecture group can have full control over what artifacts can be referenced by a large development team.

9.4 Accessing Artifact Procurement

If you installed Nexus Professional, the Nexus Procurement Suite is already installed. Just start Nexus and look for the Artifact Procurement option in the left-hand Enterprise navigation menu of the Nexus interface.

9.5 Setting up a Procured Repository

This section will walk through the process of creating and configuring a hosted repository named "Procured Central" which will be procured from the "Maven Central" proxy repository. Setting up a procured repository consists of the following steps:

- Enabling Remote Index Downloads for a Proxy Repository
- Creating a Hosted Repository to be Procured
- Configuring Procurement for the Hosted Repository
- Configuring Procurement Rules

Before configuring a procured repository, you need to make sure that you have enabled Remote Index downloading for the proxied repository that will serve as the source for your procured repository.

Note

If you are attempting to procure artifacts from a remote repository which does not have a repository index, you can still use the procurement suite. Without a remote repository index, you will need to configure procurement rules manually without the benefit of the already populated repository tree shown in Section 9.6.

9.5.1 Enable Remote Index Downloads

When you configure procurement rules for a hosted repository, the administrative interface displays the repository as a tree of groups and artifacts populated from the Remote Repository's Nexus Index. Nexus ships with a set of proxy repositories, but remote index downloading is disabled by default.

To use procurement, you will need to tell Nexus to download the remote indexes for a proxy repository. Click on "Repositories" under Views/Repositories in the Nexus menu, then click on the Maven Central repository in the list of repositories. Click on the Configuration tab, locate Download Remote Indexes, and switch this option to "True" as shown in Figure 9.3.

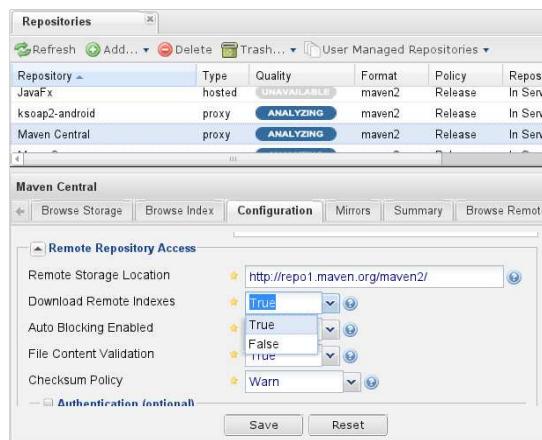


Figure 9.3: Enabling Remote Index Downloads for a Proxy Repository

Click on the Save button in the dialog shown in Figure 9.3. Right-click on the repository row in the Repositories list and select "Update Index". Nexus will then download the remote repository index and recreate the index for any Repository Groups that contain this proxied repository.

Nexus may take a few minutes to download the remote index for a large repository. Depending on your connection to the Internet, this process can take anywhere from under a minute to a few minutes. The size of the remote index for Maven Central is on the order of 30 MB.

To check on the status of the remote index download, click on System Feeds under Views in the Nexus menu. Click on the last feed to see a list of "System Changes in Nexus". If you see a log entry like the one highlighted in Figure 9.4, Nexus has successfully downloaded the Remote Index from Maven Central.

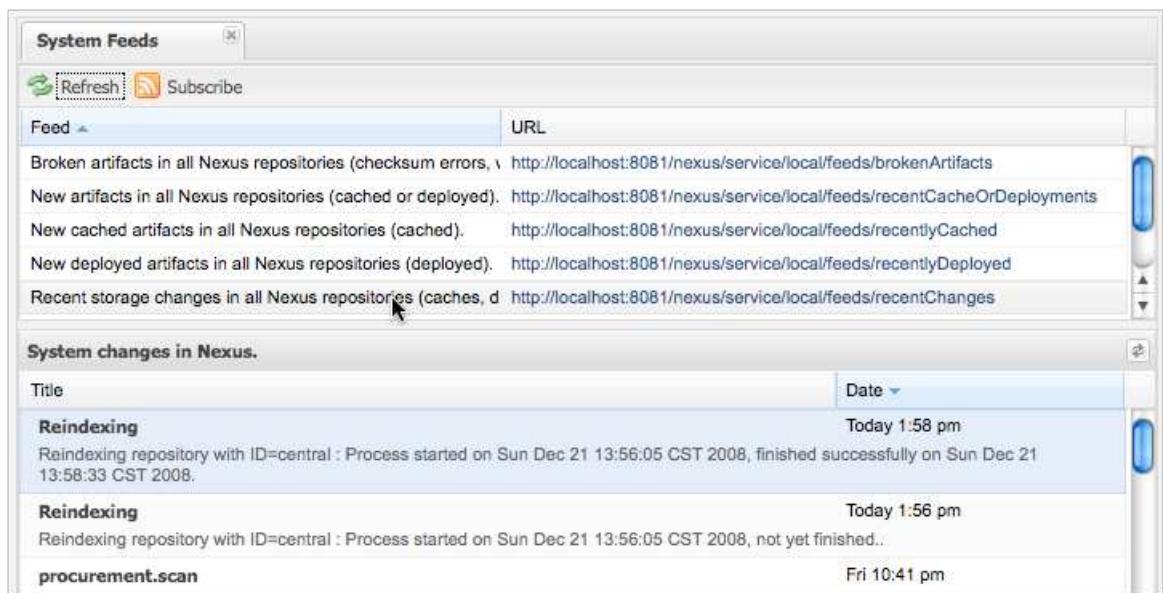


Figure 9.4: Verification that the Remote Index has been Downloaded

9.5.2 Create a Hosted Repository

When you configure procurement you are establishing a relationship between a Proxy Repository and a Hosted Repository. To create a Hosted Repository, select Repositories from the Views/Repositories section of the Nexus menu, and click on the Add button selecting Hosted Repository as shown in Figure 9.5.

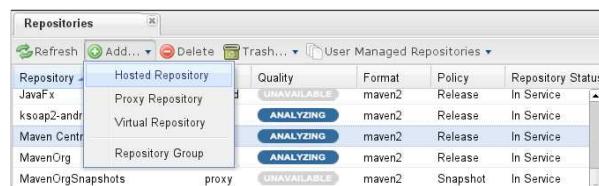


Figure 9.5: Adding a Hosted Repository

Selecting Hosted Repository will then load the Configuration form shown in Figure 9.6. Create a repos-

itory with a Repository ID of "secured" and a name of "Secured". Make the release policy "Release". Click the Save button to create the new Hosted Repository and view the new repository created displayed in Figure 9.6.

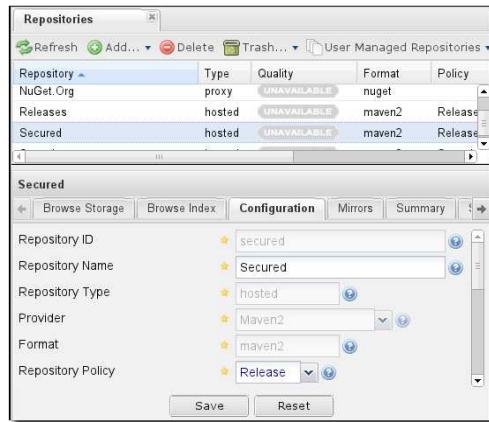


Figure 9.6: Adding the "Secured" Hosted Repository

9.5.3 Configuring Procurement for Hosted Repository

At this point, the list of Repositories will have a new Hosted repository named "Secured". The next step is to start procurement for the new Secured repository. When you do this, you are establishing a relationship between the new Hosted Repository and a Proxy Repository. In this case, we're configuring procurement for the Secured repository and we're telling the Procurement Suite to procure artifacts from the Maven Central proxy repository. To configure this relationship and to start procurement, click on Artifact Procurement under the Enterprise menu. In the Procurement panel, click on Add Procured Repository as shown in Figure 9.7.

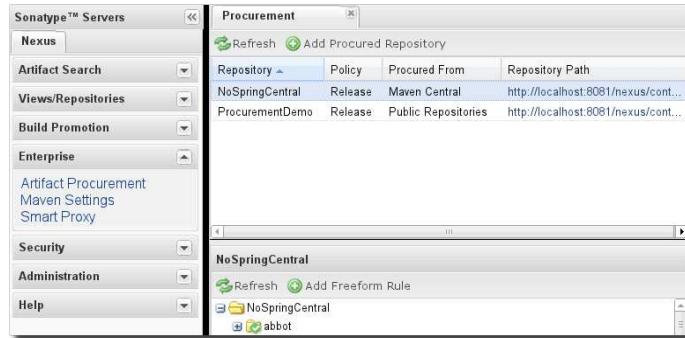


Figure 9.7: Adding a Procured Repository

You will then be presented with the Start Procurement dialog as shown in Figure 9.8. Select the "Maven Central" proxy repository from the list of available Source repositories.

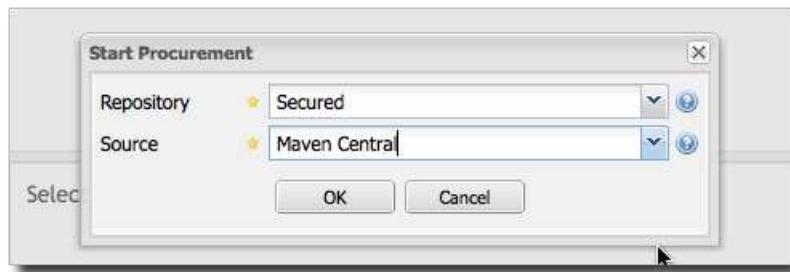


Figure 9.8: Configuring Procurement for a Hosted Repository

Procurement is now configured and started, if you are using an instance of Nexus installed on localhost port 8081, you can configure your clients to reference the new Secured repository at <http://localhost:8081/nexus/content/repositories/secured>

By default, all artifacts are denied and without further customization of the procurement rules no artifacts will be available in the new Secured repository.

One interesting thing to note about the "Secured" repository is that the Repository Type changed, once procurement was started. When procurement is activate for a Hosted repository, the repository will not show up in the Repositories list as a User Managed Repository. Instead it will show up as a Proxy Repository

as a Nexus Managed Repository. Use the drop down for User Managed/Nexus Managed Repositories in the Repositories list. Click Refresh in the Repositories list, and look at the Secured repository in the list of Nexus Managed Repositories. You will see that the repository type column contains "proxy" as shown in Figure 9.9. When procurement is started for a hosted repository it is effectively a proxy repository, and when it is stopped it will revert back to being a normal hosted repository.

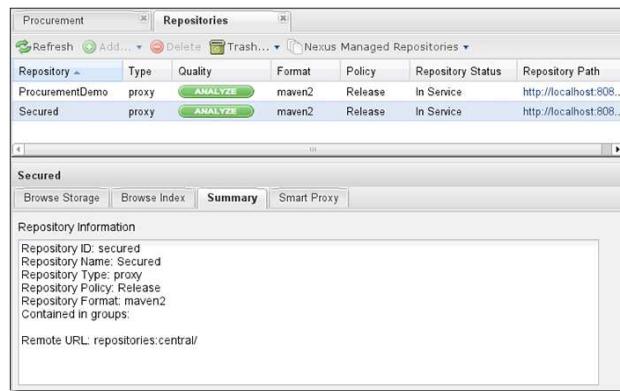


Figure 9.9: Hosted Repository is a Nexus Managed Proxy Repository while Procurement is Active

9.5.4 Procured Repository Administration

Once you've defined the relationship between a Hosted repository and a Proxy repository and you have started procurement, you can start defining the rules which will control which artifacts are allowed in a procured repository and which artifacts are denied. You can also start and stop procurement. This section details some of the administration panels and features which are available for a procured repository.

A procurement rule is a rule to allow or deny the procurement of a group, artifact, or a collection of groups or artifacts. You load the Artifact Procurement interface by selecting Artifact Procurement under the Enterprise section of the Nexus menu. Clicking on this link will load a list of procured repositories. Clicking on the repository will load the entire repository in a tree as shown in Figure 9.10.

This section will illustrate the steps required for blocking access to an entire artifact and then selectively allowing access to a particular version of that same artifact. This is a common use-case in organizations which want to standardize on specific versions of a particular dependency.

Note

If you are attempting to procure artifacts from a remote repository which does not have a repository index, you can still use the procurement suite. Without a remote repository index, you will need to configure procurement rules manually without the benefit of the already populated repository tree shown in this section.

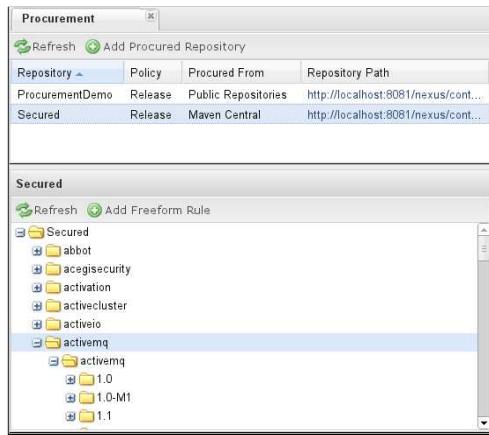


Figure 9.10: Viewing a Repository in the Artifact Procurement Interface

The directory tree in Figure 9.10 is the index of the proxy repository from which artifacts are being procured.

9.6 Configuring a Procurement Rule

To configure a procurement rule, right click on a folder in the tree. Figure 9.11 displays the procurement interface after right clicking on the jython artifact folder. In this dialog, we are telling the procurement interface to deny everything in the jython group and its sub-groups. If you attempt to retrieve any version under the jython artifact after this rule is created, the Procurement Suite will prevent access to this artifact. For example, if you attempt to build a project that depends on jython:jython:2.1, it will not be made available via the Secured repository created in Section 9.5.

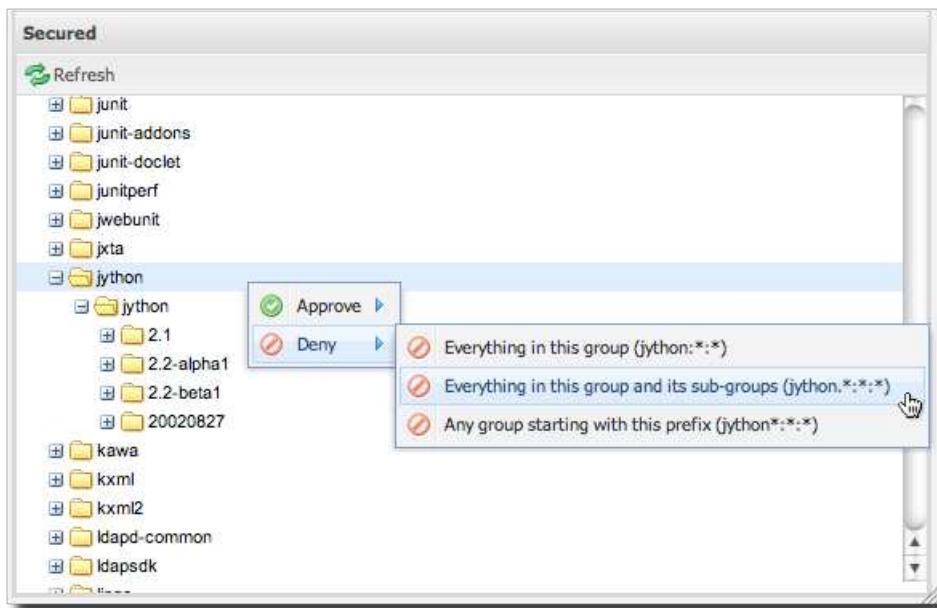


Figure 9.11: Denying Procurement for Everything Under a Group

After denying access to the entire `python` artifact, right-click on the `2.1` version folder under the `python` artifact folder. Select "Exactly the artifact" as shown in Figure 9.12 to allow for the procurement of only the `2.1` version. This has the effect of creating a specific rule which allows the procurement of a single version of the `python:python` artifact.

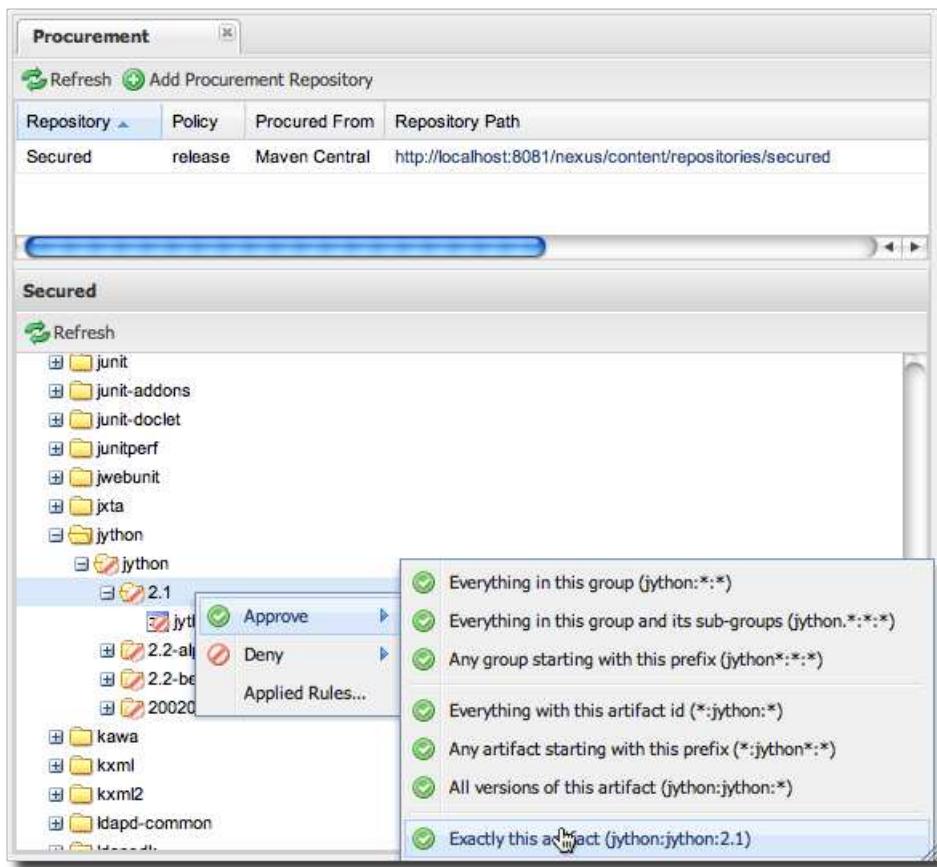


Figure 9.12: Allowing Access to a Single Artifact in a Denied Group

After allowing this element, your procurement tree is going to contain red and green markers on the affected folders as shown in Figure 9.13. While the jython artifact folder has a red slash through it the 2.1 version folder has a green check. You have configured the Procurement Suite to deny access to all versions of the jython artifact except version 2.1.

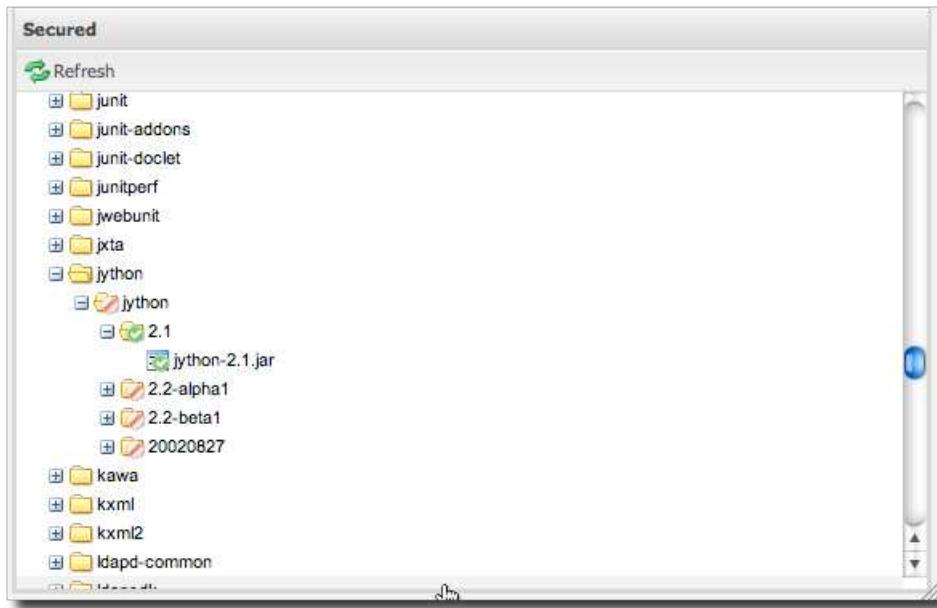


Figure 9.13: Viewing the Effect of Composite Procurement Rules on the Tree

9.7 Managing Procurement Rules

Once you've created a set of procurement rules you are going to want to know how to view the procurement rules which are applicable to a particular node. Procurement rules will frequently overlap; for example, in Section 9.5.4, there are two rules: one rule applies to all of the versions (folder) below the jython artifact folder, and the other rule applies to a particular version directory. Since rules can overlap, Nexus provides a simple way to list and manage the rules which apply to a specific node in the directory tree. Figure 9.13 shows the resolved procurement rules after clicking on the Jython 2.1 version artifact from Section 9.5.

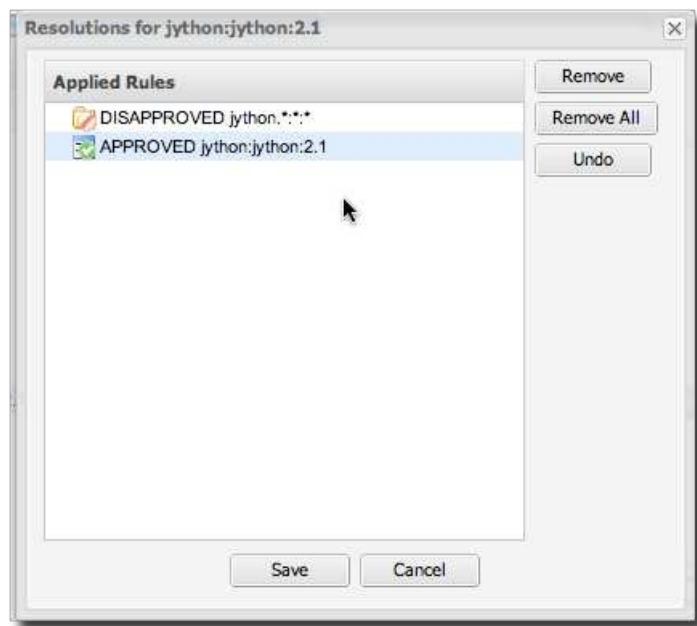


Figure 9.14: Effective Procurement Rules for a Particular Node

This dialog gives the procurement administrator a fine-grained view into the rules that apply to a particular node. From this dialog you can remove rules which apply to a specific node.

9.8 Stopping Procurement

Some organizations may want to "lock down" the artifacts that a release build can depend upon, and it is also a good idea to make sure that your build isn't going to be affected by changes to a repository not under your control. A procurement administrator might configure a procured repository, start procurement, and run an enterprise build against the repository to populate the procured, hosted repository with all of the necessary artifacts. After this process, the procurement administrator can stop procurement and continue to run the same release build against the hosted repository which now contains all of the procured artifacts.

To stop procurement, go to the Artifact Procurement management interface by clicking on Artifact Procurement under the Enterprise section of the Nexus menu. Right click on the repository and choose Stop Procurement as shown in Figure 9.15.

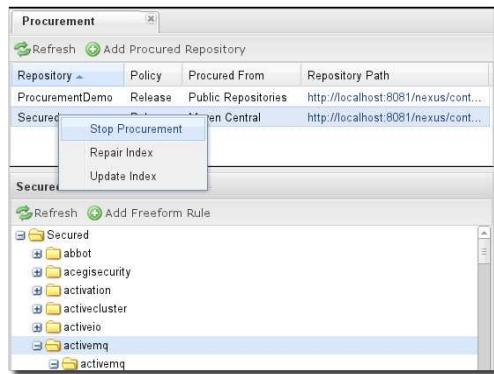


Figure 9.15: Stopping Procurement for a Procured Repository

After choosing Stop Procurement, you will then see a dialog confirming your decision to stop procurement. Once procurement is stopped, the Secure repository will revert back to being a plain-old Hosted Repository.

Chapter 10

Build Promotion with the Nexus Staging Suite

10.1 Introduction

If you release software, you will often need to test a release before deploying it to a production system or an externally accessible repository. For example, if you are developing a large, enterprise web application you may want to stage a release candidate to a production system and perform a series of rigorous tests before a release manager makes a decision to either return a system to development or deploy a system to production.

The Nexus Staging Suite in Nexus Professional allows an organization to create a temporary staging repository and to manage the promotion of artifacts from a staging repository to a release repository. This ability to create an isolated, release candidate repository that can be discarded or promoted makes it possible to support the decisions that go into certifying a release.

10.1.1 Releasing Software without a Staging Repository

Without the Staging Suite, when a developer deploys an artifact to a Hosted repository such as the Release repository, this artifact is published to a hosted repository and is immediately made available - there is no

oversight, there is no approval or certification process. There is no chance to test the artifact before writing the artifact to a hosted repository. If there is a mistake in the release, often the only option available is to republish the artifacts to the release repository or deploy a new version of the artifacts.

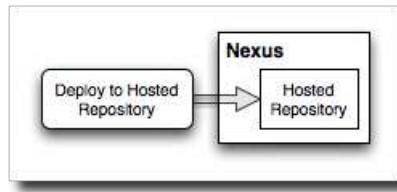


Figure 10.1: Without the Nexus Staging Suite

While this is acceptable for some users, organizations and enterprises with a QA cycle often need a temporary staging repository for potential release candidates: a staging repository. With the Nexus Staging Suite, an organization can automatically stage releases to a temporary repository which can then be used to test and certify a set of artifacts before they are published to a final release repository. This temporary repository can then be promoted as a whole or dropped depending on the results of testing.

10.1.2 How the Staging Suite Works

Here's how staging works in Nexus Professional:

1. A developer deploys an artifact (or a set of artifacts) to Nexus Professional.
2. The Staging Suite intercepts this deployment and matches the artifact's path against a set of Staging Profiles.
3. If the path of the artifact activates a staging profile, a temporary staging repository is created and the artifacts are deployed to this repository.
4. Once the developer has deployed a set of artifacts to Nexus, they will then "Close" the staging repository.
5. The Staging Suite will then add this temporary staging repository to one or more Target Repository Groups.

Once the staging repository is closed, and the staging repository has been added to a Target repository group, the artifacts in the staging repository are then made available to developers or testers via a reposi-

tory group. Tests can be performed on the artifacts as if they were already published in a hosted repository. From this point, one of two things can happen to a staging repository:

Release

A Nexus user can "release" a staging repository and select a hosted repository to publish artifacts to. Releasing the contents of a repository publishes all artifacts from the staging repository to a hosted repository and deletes the temporary staging repository.

Drop

A Nexus user can "drop" a staging repository. Dropping a staging repository will remove it from any groups and delete the temporary staging repository.

Promote

If your Nexus installation contains Build Promotion profiles, you will also see an option to "promote" a staging repository to a Build Promotion Group. When you promote a staging repository you can expose the contents of that staging repository via additional groups. Build Promotion profiles are explained in detail in the next section.

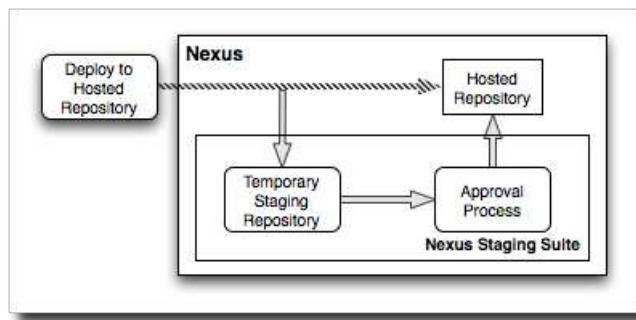


Figure 10.2: With Nexus Staging Suite

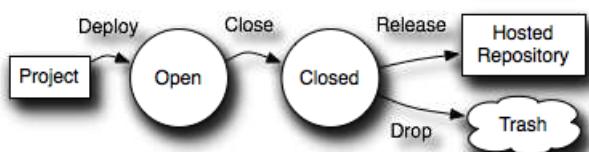


Figure 10.3: Supplying a Description for a Staging Release

10.1.3 Multi-level Staging and Build Promotion

Nexus Professional also supports multi-level staging and build promotion. With multi-level staging, a staging repository can be tested and then promoted to a separate "build promotion" profile and exposed through different repository groups to allow for additional testing and qualification before a final release. Figure 10.4 illustrates a potential use for multi-level staging:

Stage

A developer publishes artifacts to a QA staging profile which exposes the staged artifacts in a QA repository group used by an internal quality assurance team for testing.

Promote to Beta

Once the QA team has successfully completed testing, they promote the temporary staging repository to build promotion profile which will expose the staged artifacts to a limited set of customers who have agreed to act as a beta testers for a new feature.

Release

Once this closed beta testing period is finished, the staged repository is then released and the artifacts it contains are published to a hosted release repository and exposed via the public repository group.

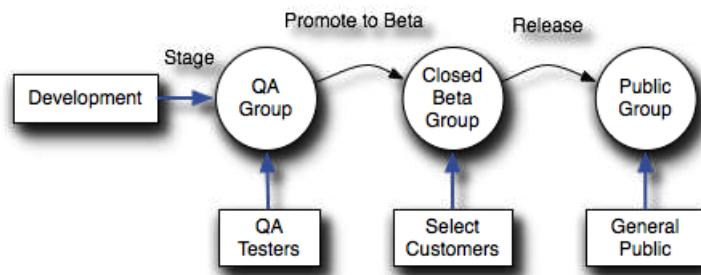


Figure 10.4: Multi-level Staging and Build Promotion

To support this multi-level staging feature, you can configure Build Promotion profiles as detailed in Section 10.2.2. Once you have promoted a Staging Repository to a Build Promotion profile, you can drop, promote, or release the artifacts it contains as detailed in Section 10.2.

10.2 Using the Nexus Staging Suite

To use the Staging Suite in Nexus Professional, start Nexus and look for the Staging Profiles, Staging Repositories, Staging Rulesets, and Staging Upload options in the left-hand navigation menu of the Nexus interface. If you see the links shown in Figure 10.5, the Staging Suite is available and ready to be configured.

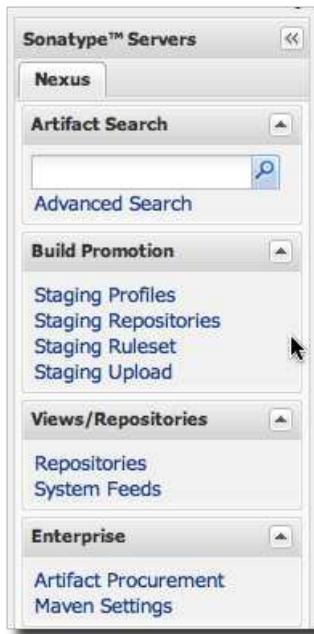


Figure 10.5: Enterprise Menu after Staging Suite Installation

10.2.1 Configuring Staging Profiles

Staging Profiles define the rules by which artifact deployments are staged in Staging Repositories. Staging Repositories are created as they are needed and are the primary mechanism by which Nexus users can promote or discard the contents of a staging repository to a hosted repository. A staging profile uses a Repository Target to match artifacts as they are deployed. If a matching artifact is deployed to Nexus, the Staging Suite will intercept this deployment and store the artifact in a staging repository.

The process for configuring a new Staging Profile is as follows:

1. Configure a Repository Target to match artifacts under the groupId you will be deploying artifacts to. If you are releasing all of your software under the groupId com.example, you would configure a target that matches the pattern `./com/example/.`.
2. Create a new Staging Profile using the target defined in the previous step. When you configure this staging profile, you will be defining a target repository group. When the Staging Suite intercepts an artifact and places it in a staging repository, this staging repository will be added to the specified target repository group.
3. Assign the appropriate Staging-specific roles to the appropriate users. When you create a Staging Profile, Nexus also creates two new roles that grant access and privileges to the repositories created by this Staging Profile.

The following sections provide a more detailed look at the process of configuring a single staging profile in Nexus Professional.

Configuring a Repository Target

The Staging Suite intercepts deployments to repository targets. For example, if you wanted to intercept all deployments to the com.sonatype.sample groupId, you would create a Repository Target called the "Sample Target" with a pattern expression of `/com/sonatype/sample/.`. Do this by clicking on "Repository Targets" in the "Security" section of the left-hand navigation menu in Nexus and then clicking on the Add button.

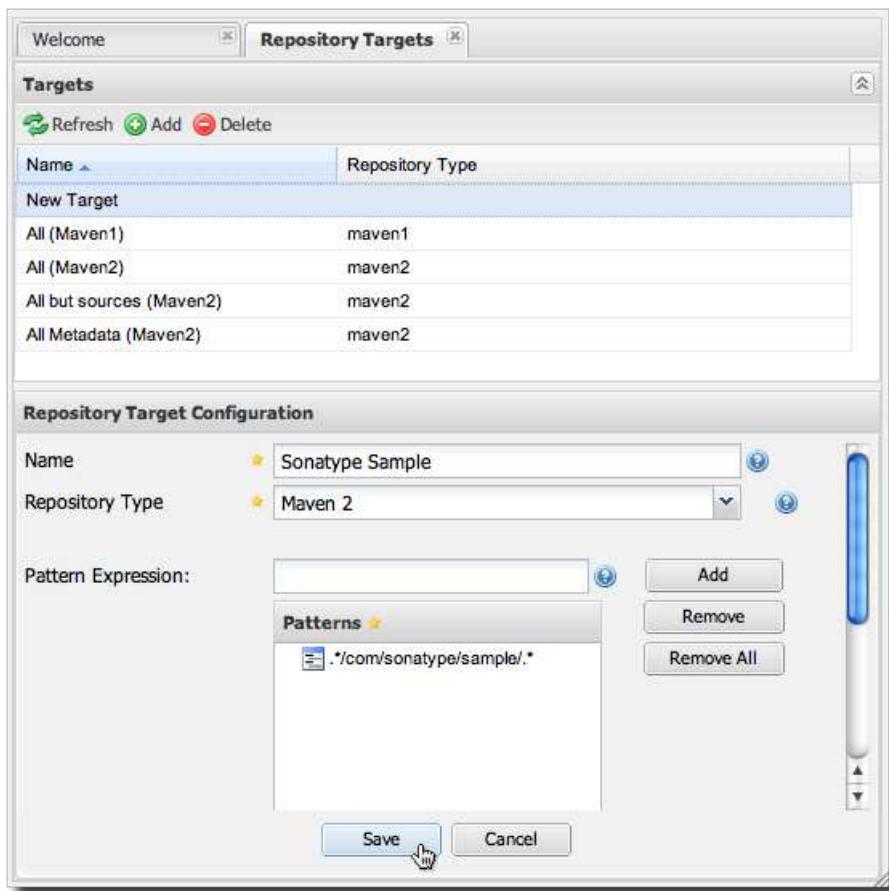


Figure 10.6: Adding a Repository Target for com.sonatype.sample

Configuring Staging Profiles

Staging profiles control the process by which artifacts are selected for staging. When you define a Staging profile, you are defining a set of rules which will control the way in which Nexus intercepts an artifact deployment. When you click on Staging Profiles in the Nexus menu, you will see a list of configured staging profiles. Clicking on Add... will display the drop-down menu shown in Figure 10.7.



Figure 10.7: Multi-level Staging and Build Promotion

Selecting Staging Profile will create a new Staging Profile and display the form shown in Figure 10.8.

Figure 10.8 defines a Staging Profile using the Repository Target. This target will match all artifacts under the com.sonatype.sample groupId (or the com/sonatype/sample repository path). This staging profile uses the "Maven2 (hosted, release)" as a template for newly created temporary staging repositories, and it will automatically add closed staging repositories to the Public Repositories group.

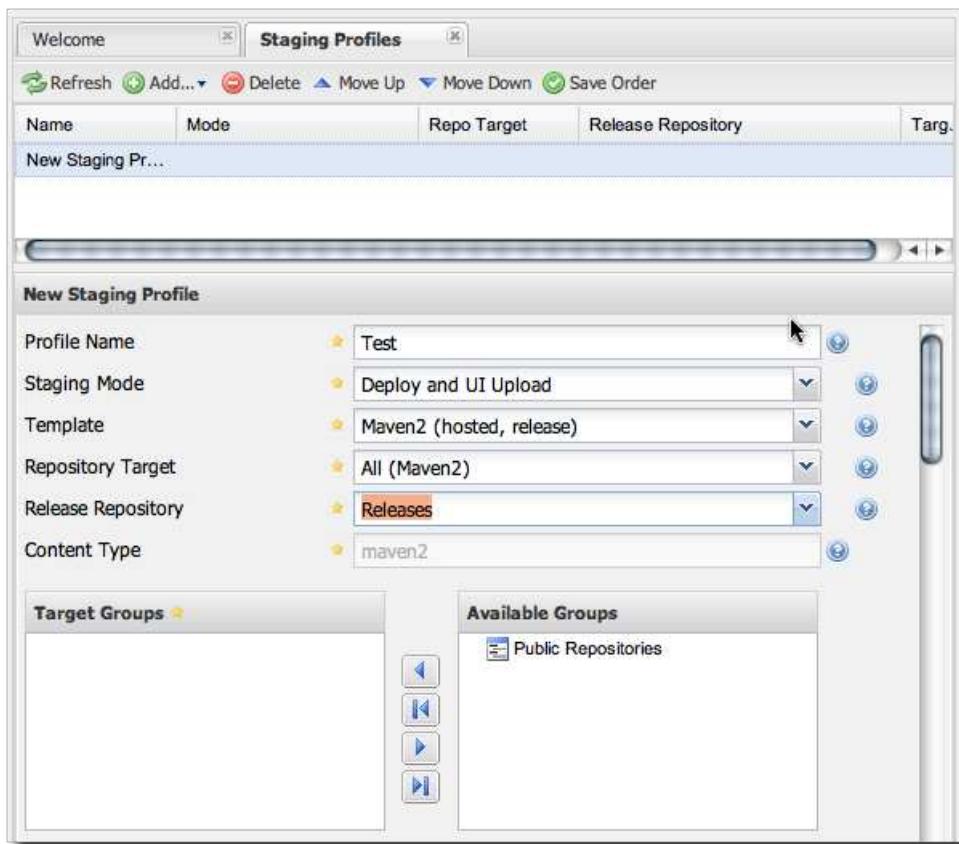


Figure 10.8: Editing a Staging Profile

This form allows you to configure a profile. Every profile has a name, is associated with a Repository Target, and points to a template to use when creating a new staging repository. The Staging Profile configuration panel contains the following fields:

Profile Name

The Name of the Staging Profile. This can be an arbitrary value. It is simply a convenience for the Nexus Administrator, and it is also used to create Nexus roles that are used to grant permissions to view and manipulate staging repositories created by this profile.

Staging Mode

This field contains the options "Deploy", "UI Upload", and "Deploy and UI Upload". This controls how artifacts can be staged to this staging profile. If Deploy is selected, artifacts can only be

deployed using Maven to upload build artifacts. If UI Upload is selected, users can upload artifacts to Nexus using the Nexus user interface.

Template

Defines a template for the temporary staging repository. The current version of Nexus Professional provides with the options "Maven2 (hosted, release)" and "Maven1 (hosted, release)".

Repository Target

This is a reference to the target which we defined previously. When a developer deploys an artifact to the Staging URL, the Staging Suite will check to see if the artifact matches the patterns defined in the Repository Target. The Target defines the "trigger" for the creation of a Staging Repository.

Release Repository

Staged artifacts are stored in a temporary staging repository which is made available via Target Groups. Once a staged deployment has been successfully tested, artifacts contained in the temporary staging repository are promoted to a hosted repository. The Release Repository setting configures the target release repository for this staging profile.

Content Type

Nexus can create staging repositories for repositories of type Maven1, Maven2, and P2 repositories. This value is automatically selected based on the chosen template. This chapter only deals with Maven2 repository types.

Target Groups

When a Staging Repository is "closed" and is made available to users and developers involved in the testing process, the temporary Staging Repository is added to one or more Repository Groups. This field defines those groups.

Close Repository Notification Settings

After a developer has deployed a set of related release artifacts, a staging repository is "closed". This means that no further artifacts can be deployed to the same staging repository. A repository would be closed when a developer is satisfied that a collection of staged artifacts is ready to be certified by a manager or a quality assurance resource. In this setting, it is possible to define email addresses and Roles which should be notified of a staging repository being closed. A notification email will be sent to all specified email addresses, as well as all Nexus users in the specified roles, informing that a staging repository has been closed. It is also possible to select that the creator of the staging repository receives this notification.

Promote Repository Notification Settings

Once a closed staging repository has been certified by whoever is responsible for testing and checking a staged release, it can then be promoted (published) or dropped (discarded). In this setting, it is possible define email addresses and Roles which should be notified of a staging repository being promoted. A notification email will be sent to all specified email addresses, as well as all Nexus users in the specified roles, informing that a staging repository has been promoted. It is also possible to select that the creator of the staging repository receives this notification.

Drop Repository Notification Settings

In this setting, it is possible define email addresses and Roles which should be notified of a staging

repository being dropped. A notification email will be sent to all specified email addresses, as well as all Nexus users in the specified roles, informing that a staging repository has been dropped. It is also possible to select that the creator of the staging repository receives this notification.

Close Repository Staging Rulesets

This defines the rulesets which will be applied to a staging repository before it can be closed. If the staging repository does not pass the rules defined in the specified rulesets, you will be unable to close it. For more information about rulesets, see Section [10.5](#).

Promote Repository Staging Rulesets

This defines the rulesets which will be applied to a staging repository on promotion. If the staging repository does not pass the rules defined in the specified rulesets, the promotion will fail with an error message supplied by the failing rule. For more information about rulesets, see Section [10.5](#).

Once you've created a Staging Repository with the values shown in Figure [10.8](#), you are ready to perform a test deployment to the Staging URL.

10.2.2 Configuring Build Promotion Profiles

A Build Promotion profile is used when you need to add an additional step between initial staging and final release. To add a new Build Promotion profile, open the Staging Profiles link from the Nexus menu and click on Add... to display the drop-down menu shown in Figure [10.9](#). Select Build Promotion Profile from this drop-down to create a new Build Promotion Profile.



Figure 10.9: Multi-level Staging and Build Promotion

After creating a new Build Promotion profile, you will see the form shown in Figure [10.10](#). This form contains the following configuration fields:

Profile Name

This is the name for the Build Promotion profile which will be displayed in the promotion dialog shown in Figure 10.26. This name will also be associated with repositories created from this promotion profile.

Template

This is the template for repositories generated by this Build Promotion profile. The default value for this field is "Maven2 (group)".

Target Groups

This is the most important configuration field for a Build Promotion profile. It controls the group that promoted artifacts will be made available through. Artifacts can be made available through one or more groups.

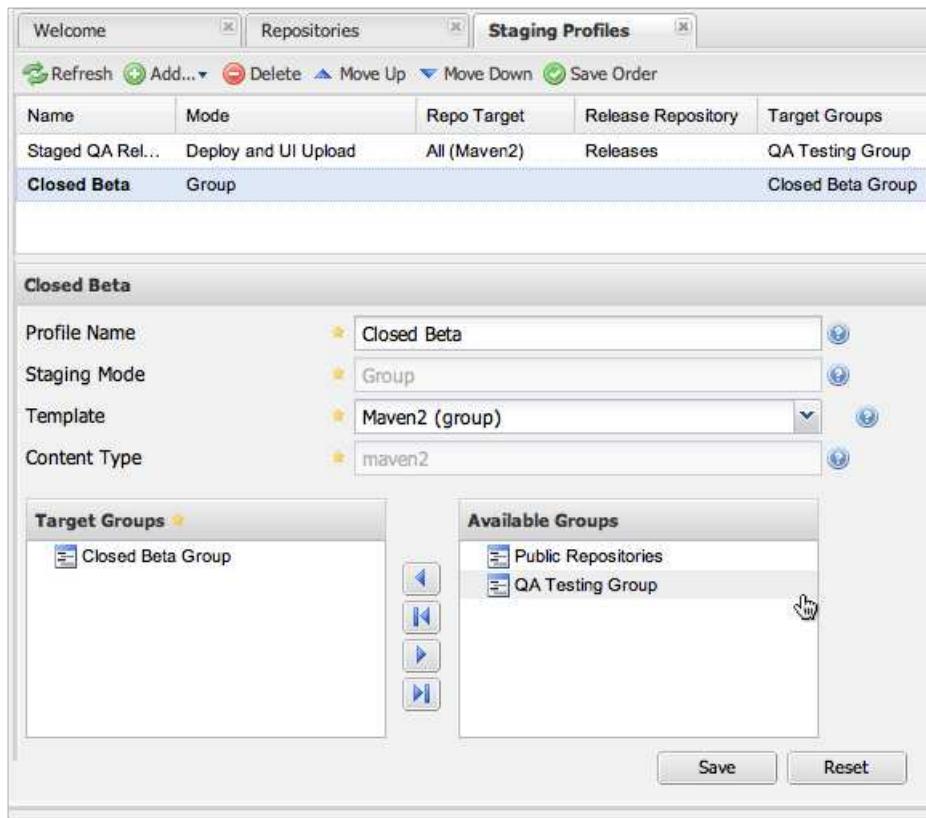


Figure 10.10: Configuring a Build Promotion Profile

10.2.3 Adding the Staging Deployer Role

To perform a staged deployment, the user deploying the artifact must have the "Staging: Deployer (admin)" role or the "Staging: Deployer" role for a specific Staging Profile.

When you create a Staging Profile, Nexus will create two new Nexus roles that grant permissions specific to that staging profile. If you created the Staging profile from the previous section, Nexus would have created two roles:

Staging: Repositories (Release Staging Profile)

This role grants a user read and view access to the staging repositories created by a specific staging profile.

Staging: Deployer (Release Staging Profile)

This role grants all of the privileges from the Staging: Repositories role and it grants the user permission to deploy artifacts, close a staging repository, and promote or drop a staging repository created by a specific staging profile.

In addition to the profile-specific staging roles, the Staging Suite also defines two universal roles which grant read-only or deployer rights across all staging repositories. These roles are:

Staging: Repositories (admin)

This role grants a user read and view access to all staging repositories.

Staging: Deployer (admin)

This role grants a user all of the privileges from the Staging: Repositories role and it grants the user permission to deploy artifacts to any staging repository, close all staging repositories, and promote or drop all staging repositories.

To configure the deployment user with the appropriate staging role, click on Users under the Security menu in the Nexus menu. Once you see the Users panel, click on the deployment user to edit this user's roles. If the Staging Suite is installed, you should see the "Staging: Deployer (admin)" role listed in Available Roles. Select the "Staging: Deployer (admin)" role and then click the left arrow to add this role to the deployment user's list of assigned roles.

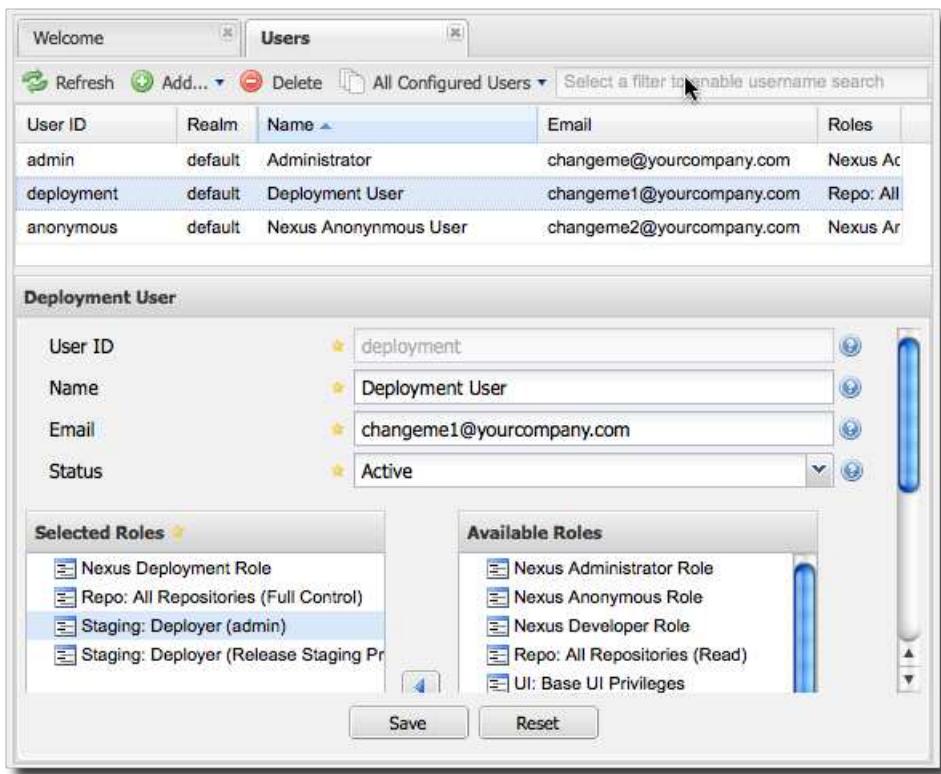


Figure 10.11: Assigning the Staging Deployer Role to the deployment user

Once the deployment user has the "Staging: Deployer (admin)" role, you can then use this user to deploy to the staging URL and trigger any Staging Profile. Without this permission, the deployment user would not be able to publish a staged artifact.

If you need to add a specific permission to activate a single Staging Profile, you would select that specific role in the Available Roles list shown in Figure 10.11. In this figure, note that there are two "Staging: Deployer" roles: one for general administrative permission to deploy to any staging profile, and another which targets a specific staging profile.

10.3 Performing a Staged Deployment with Maven

In the previous section, you created a Staging Profile which references the Repository Target created previously. If the Staging Suite is configured correctly, any deployment to the staging URL under the groupId com.sonatype.sample should be intercepted by the Staging Suite and placed in a temporary staging repository. Once this repository has been closed, it will be made available in the Target Group you selected when you configured the Staging Profile.

In this section, you will create a new project using the Maven Archetype plugin to test the Staging Profile you created in the previous section.

10.3.1 Creating a New Project

To create a new project run mvn archetype:generate. Running this at the command line will bring up a list of archetypes, choose the default maven-archetype-quickstart, and use the identifier values listed in Table 10.1 for the new project.

Table 10.1: Identifiers for New Project

Identifier	Value
groupId	com.sonatype.sample
artifactId	staging-test
version	1.0
package	com.sonatype.sample

If the archetype generate goal is executed successfully, you should have output which resembles the following screen listing:

```
$ mvn archetype:generate  
...  
[INFO] [archetype:generate]  
[INFO] Generating project in Interactive mode  
[INFO] No archetype defined. Using maven-archetype-quickstart \  
(org.apache.maven.archetypes:maven-archetype-quickstart:1.0)  
Choose archetype:  
1: internal -> appfuse-basic-jsf (AppFuse archetype for creating a \  
...
```

```
web application with Hibernate, Spring and JSF)
...
41: internal -> gmaven-archetype-mojo (Groovy mojo archetype)
Choose a number: (1/.../41) 16: : 16
Define value for groupId: : com.sonatype.sample
Define value for artifactId: : staging-test
Define value for version: 1.0-SNAPSHOT: : 1.0
Define value for package: com.sonatype.sample: : com.sonatype.sample
Confirm properties configuration:
groupId: com.sonatype.sample
artifactId: staging-test
version: 1.0
package: com.sonatype.sample
Y: :
[INFO] Parameter: groupId, Value: com.sonatype.sample
[INFO] Parameter: packageName, Value: com.sonatype.sample
[INFO] Parameter: basedir, Value: /private/tmp
[INFO] Parameter: package, Value: com.sonatype.sample
[INFO] Parameter: version, Value: 1.0
[INFO] Parameter: artifactId, Value: staging-test
...
[INFO] BUILD SUCCESSFUL
```

10.3.2 Update the POM Deployment Configuration

To deploy a staged release, a developer needs to deploy to the staging URL. To configure this new project to deploy to the Staging URL, add the a distributionManagement element to the stage-test project's POM.

Listing the Staging URL in distributionManagement

```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.sonatype.sample</groupId>
  <artifactId>staging-test</artifactId>
  <packaging>jar</packaging>
  <version>1.0</version>
  <name>staging-test</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
```

```
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
<distributionManagement>
  <repository>
    <id>nexus</id>
    <name>Nexus Staging Repo</name>
    <url>http://localhost:8081/nexus/service/local/staging/deploy/maven2 <-
      /</url>
  </repository>
</distributionManagement>
</project>
```

This configuration element, distributionManagement, defines the repository to which our deployment will be made. It references the Staging Suite's Staging URL: <http://localhost:8081/nexus/service/local/-staging/deploy/maven2>

This URL acts as a something of a virtual repository to be published to. If an artifact being published matches one of the Repository Targets in a Staging Profile, that Staging Profile is "activated" and a temporary Staging Repository is created for a specific client as defined by the combination of a client's IP address, Deployment User name, and User-Agent.

10.3.3 Update settings.xml with Deployment Credentials

To successfully deploy to your Nexus instance, you will need to update your Maven Settings with the credentials for the deployment user. These credentials are stored in the Maven Settings file in `~/.m2/settings.xml`.

To add these credentials, add the following element to the servers element in your `~/.m2/settings.xml` file as shown in [Listing deployment credentials in Maven Settings](#).

Listing deployment credentials in Maven Settings

```
<settings>
  ...
  <servers>
    ...
    <server>
      <id>nexus</id>
      <username>deployment</username>
      <password>deployment123</password>
```

```
</server>
</servers>
...
</settings>
```

Note that the server identifier listed in [Listing deployment credentials in Maven Settings](#) matches the server identifier listed in [Listing the Staging URL in distributionManagement](#). The deployment credential listed in [Listing deployment credentials in Maven Settings](#) contains the default password for the Nexus deployment user - deployment123. You should change this password to match the deployment password for your Nexus installation.

10.3.4 Deploying to a Staged Repository

Once the sample project's distributionManagement has been set to point at the Nexus Staging URL and your deployment credentials are updated in your `~/.m2/settings.xml` file, you can deploy to the Staging URL. To do this, run `mvn deploy`

```
$ mvn deploy
[INFO] Scanning for projects...
[INFO]   ↵
-----
[INFO] Building staging-test
[INFO]   task-segment: [deploy]
[INFO]   ↵
-----
[INFO] [resources:resources]
[INFO] Using default encoding to copy filtered resources.
[INFO] [compiler:compile]
[INFO] Nothing to compile - all classes are up to date
[INFO] [resources:testResources]
[INFO] Using default encoding to copy filtered resources.
[INFO] [compiler:testCompile]
[INFO] Nothing to compile - all classes are up to date
[INFO] [surefire:test]
[INFO] Surefire report directory: /private/tmp/staging-test/target/ ↵
      surefire-reports
-----
[INFO] [jar:jar]
[INFO] [install:install]
[INFO] Installing /private/tmp/staging-test/target/staging-test-1.0.jar to ↵
      \
```

```
~/.m2/repository/com/sonatype/sample/staging-test/1.0/staging-test-1.0.jar
[INFO] [deploy:deploy]
altDeploymentRepository = null
Uploading: http://localhost:8081/nexus/service/local/staging/deploy/maven2 ↵
  \\
com/sonatype/sample/staging-test/1.0/staging-test-1.0.jar
2K uploaded
[INFO] Uploading project information for staging-test 1.0
[INFO] Retrieving previous metadata from nexus
[INFO] repository metadata for: 'artifact com.sonatype.sample:staging-test ↵
  ,
could not be found on repository: nexus, so will be created
[INFO] Uploading repository metadata for: 'artifact com.sonatype.sample: ↵
  staging-test'
[INFO] ↵
----- ↵
[INFO] BUILD SUCCESSFUL
```

10.4 Manually Uploading a Staged Deployment in Nexus

You can also upload a staged deployment via the Nexus interface. To upload a staged deployment, select Staging Upload from the Nexus menu. Clicking Staging Upload will show the panel shown in Figure 10.12.

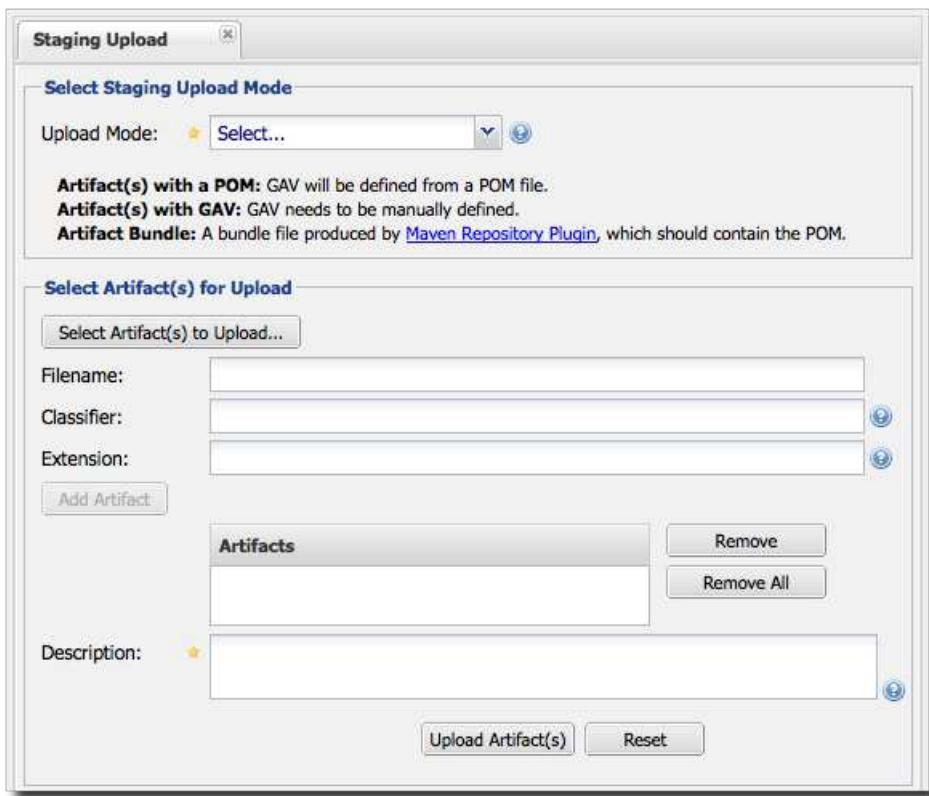


Figure 10.12: Uploading a Staged Deployment in Nexus

To upload an artifact, click on Select Artifact(s) for Upload... and select one or more artifacts from the filesystem to upload. Once you have selected an artifact, you can modify the classifier and the extension before clicking on the Add Artifact button. Once you have clicked on the Add Artifact button, you can then configure the source of the Group, Artifact, Version (GAV) parameters.

If the artifact you are uploading is a JAR file that was created by Maven it will already have POM information embedded in it, but if you are uploading a JAR from a vendor you will likely need to set the Group Identifier, Artifact Identifier, and Version manually. To do this, select GAV Parameters from the GAV Definition drop-down at the top of this form. Selecting GAV Parameters will expose a set of form fields which will let you set the Group, Artifact, Version, and Packaging of the artifacts being uploaded. If you would prefer to set the Group, Artifact, and Version from a POM file which was associated with the uploaded artifact, select From POM in the GAV Definition drop-down. Selecting From POM in this drop-down will expose a button labelled "Select POM to Upload". Once a POM file has been selected for

upload, the name of the POM file will be displayed in the form field below this button.

The Staging Upload panel supports multiple artifacts with the same Group, Artifact, and Version identifiers. For example, if you need to upload multiple artifacts with different classifiers, you may do so by clicking on Select Artifact(s) for Upload and Add Artifact multiple times. This interface also accepts an Artifact Bundle which is a JAR that contains more than one artifact.

Once a staging artifact upload has been completely configured, click on Upload Artifact(s) button to begin the upload process. Nexus will upload the artifacts to the Staging URL which will trigger any staging profiles that are activated by the upload. If a staging profile is activated, a new staging repository will be created and can be managed using the procedures outlined in Section [10.6](#).

10.5 Managing Rulesets

Nexus Professional has the ability to define staging rules that must be satisfied before a staging repository can be promoted.

10.5.1 Managing Staging Rulesets

Staging Rulesets are groups of rules that are applied to a Staging repository at promotion time. A staging repository associated with a staging ruleset cannot be promoted until all of the rules associated with the rulesets have been satisfied. This feature allows you to set standards for your own hosted repositories, and it is the mechanism that is used to guarantee the consistency of artifacts stored in the Maven Central repository.

Nexus Professional contains the following rules:

Staging Javadoc Validation

The Staging Javadoc Validation rule will verify that every project has an artifact with the javadoc classifier. If you attempt to promote a staging repository which contains artifacts not accompanied by "-javadoc.jar" artifacts, this validation rule will fail.

Staging Artifact Uniqueness Validation

This rule checks to see that the artifact being released, promoted, or staged is unique in a particular Nexus instance.

Staging Checksum Validation

This rule validates file checksums against published artifacts.

Staging No Release Repository

This rule will fail if a particular staging profile is not defined with a release repository.

Staging POM Validation

The Staging POM Validation rule will verify Project URL - project/url, Project Licenses - project/licenses and Project SCM Information - project/scm. If any of these POM elements are missing or empty, this Staging Ruleset will cause a promotion to fail.

Staging Signature Validation

The Staging Signature Validation rule verifies that every item in the repository has a valid PGP signature. If you attempt to promote a staging repository which contains artifacts not accompanied by valid PGP signature, this validation will fail.

Staging Sources Validation

The Staging Sources Validation rule will verify that every project has an artifact with the sources classifier. If you attempt to promote a staging repository which contains artifacts not accompanied by "-sources.jar" artifacts, this validation rule will fail.

To create a Staging Ruleset, click on the Staging Ruleset link in the Nexus Menu. This will load the interface shown in Figure 10.13. The Staging Ruleset panel is used to define sets of rules that can be applied to Staging Profiles. Figure 10.13 shows a ruleset which contains all four predefined staging rules.

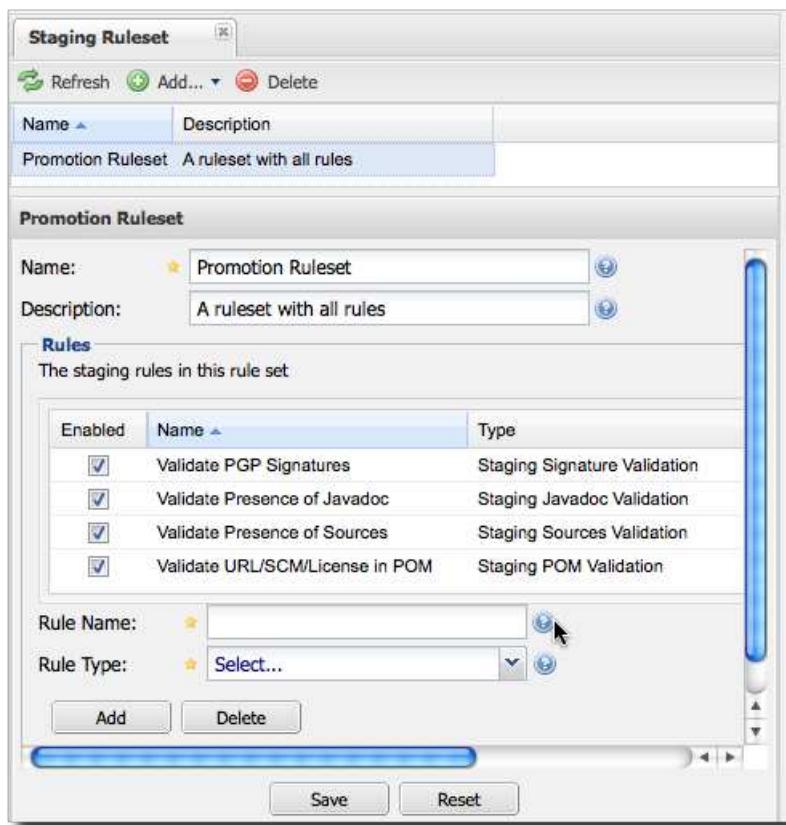


Figure 10.13: Creating a Staging Ruleset

10.5.2 Defining Rulesets for Promotion

To define a ruleset to be used for promotion, click on Staging in the Nexus menu and select a Staging Profile. Click on the Configuration tab, and scroll down to the Promote Repository Staging Rulesets section of the Staging Profile configuration as shown in Figure 10.14. The next time you attempt to promote a staging repository that was created with this profile, Nexus Professional will check that all of the rules in the associated rulesets are being adhered to.

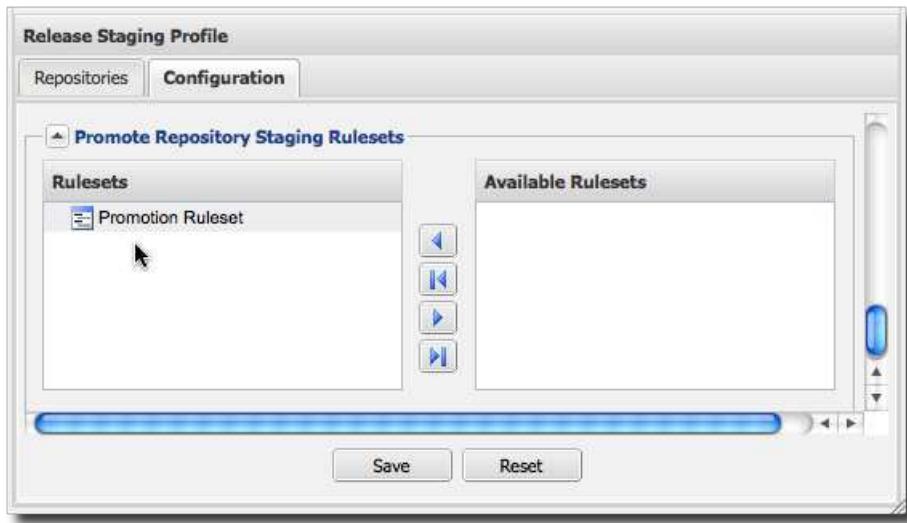


Figure 10.14: Associating a Staging Ruleset with a Staging Profile

10.6 Managing Staging Repositories in Nexus

Once you complete the process outlined in Section 10.3, you will then have an automatically generated Staging Repository. In this section, you will walk through the process of managing staging repositories. Once a staging repository has been created, there are two steps in the lifecycle of a staging repository. Once you have deployed a set of related artifacts, you must "Close" the repository moving it from an "Open" to a "Closed" state. Once a repository is in the "Closed" state it is added to a Repository Group and is made available for testing purposes. Once testing is completed, a Nexus administrator can either Promote or Drop a Closed repository.

If the repository is Dropped, the repository is discarded and removed from the Repository Group. If the repository is Promoted, the Nexus administrator can select a Hosted repository and publish the contents of the temporary staging repository to a Hosted repository.

10.6.1 Closing an Open Repository

Once you deploy an artifact that triggers a staging profile, Nexus Staging Suite will create a repository that contains the artifacts you deployed. A separate staging repository is created for every combination of User ID, IP Address, and User Agent. This means that you can perform more than one deployment to a single Staging Repository as long as you perform the deployment from the same IP, with the same deployment user, and the same installation of Maven.

You can perform multiple deployments to an "Open" staging repository, to see a list of these temporary "Open" Staging repositories, select "Staging" from the Nexus menu and click on the appropriate Staging Profile to browse a list of staging repositories which correspond to a staging profile.



Figure 10.15: Listing Repositories Associated with a Staging Profile

Once you are ready to start testing the staging repository, you will need to transition the staging repository from the "Open" state to the "Closed" state. This will close the temporary staging repository to more deployments.

To close a repository, right-click on the repository in the Staging Repositories panel and select "Close". This will bring up the following dialog for a staging deployer to describe the contents of a staging repository. This description field can be used to pass essential information to the person that needs to test a deployment.

In Figure 10.16, the description field is used to describe the release for the user that needs to certify and promote a release.

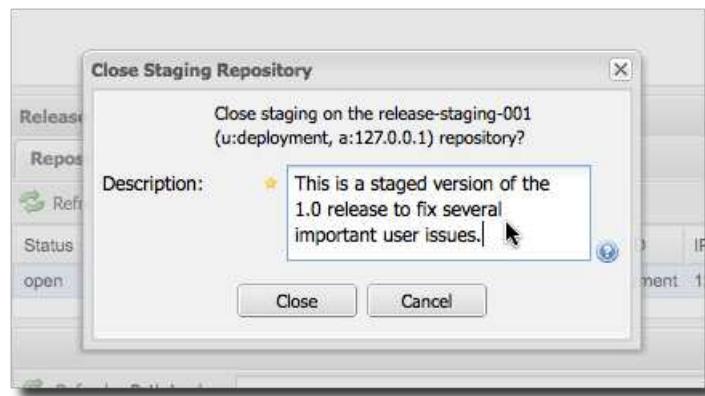


Figure 10.16: Confirmation and Description Dialog for Closing a Staging Repository

Confirming this state transition will close the repository and add the repository to a repository group. Once a repository has been closed, it will be listed as "Closed" in the Profile's Repositories tab.



Figure 10.17: Closed Repository After Selecting Finish

10.6.2 Using the Staging Repository

Once the Staging Repository has been closed, it will automatically be added to the Repository Group that was specified in the Staging Profile. Figure 10.18 shows an instance of a staging repository appended to the end of a group named "Public Repositories".

This has the effect of making the staged artifacts available to everyone who is referencing this public group. Developers who are referencing this public repository group can now test and interact with the staged artifacts as if they were published to a Hosted repository.

While the artifacts are made available in a repository group, the fact that they are held in a temporary staging directory gives the administrator the option of promoting this set of artifacts to a Hosted repository or dropping this temporary staging repository if there are problems discovered during the testing and certification process for a release.

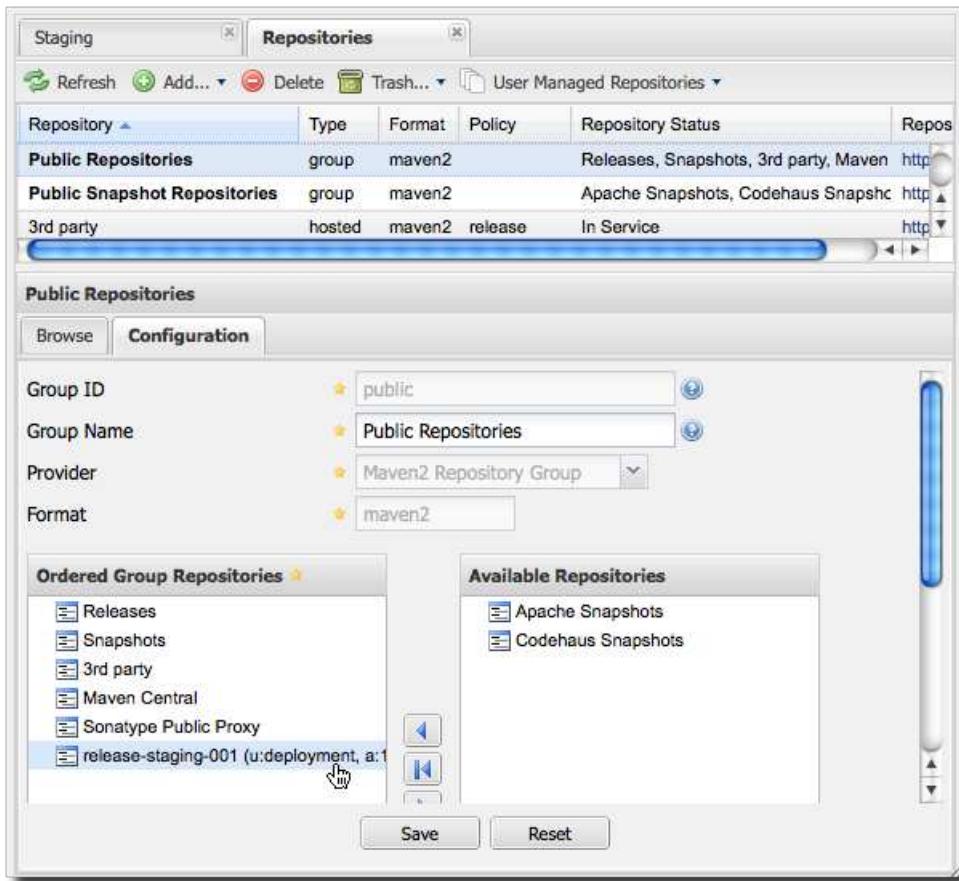


Figure 10.18: Staging Repository Added to the End of a Repository Group

Once a staging repository is closed, you can also browse and search the repository. To view Staging Repositories, click on Browse Repositories and then select Nexus Managed Repositories as shown in Figure 10.19.

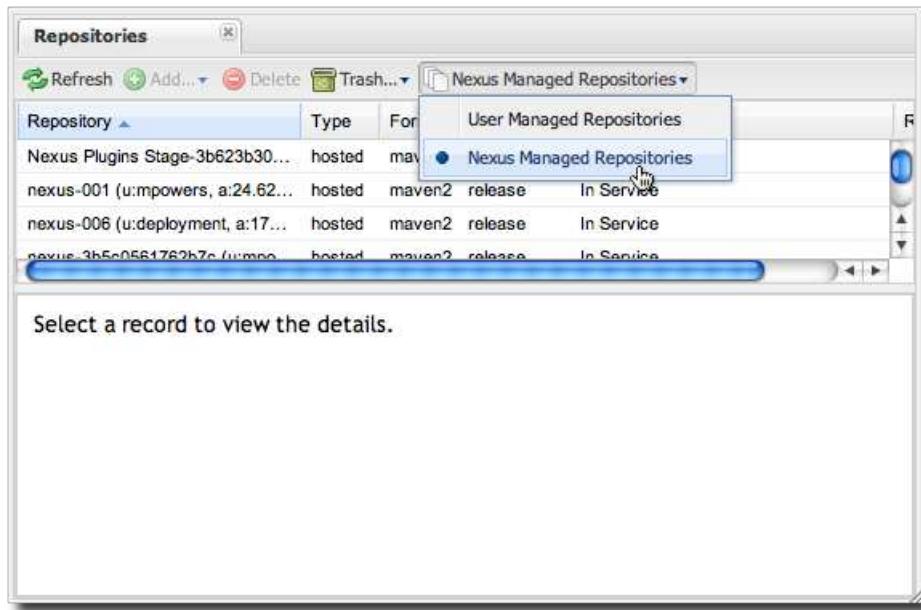


Figure 10.19: Viewing Nexus Managed Repositories

Once you've selected Nexus Managed Repositories, Nexus will then show you all of the repositories that have been created by the Nexus Staging Suite. You can select and browse this temporary Staging Repository as you would any other repository.

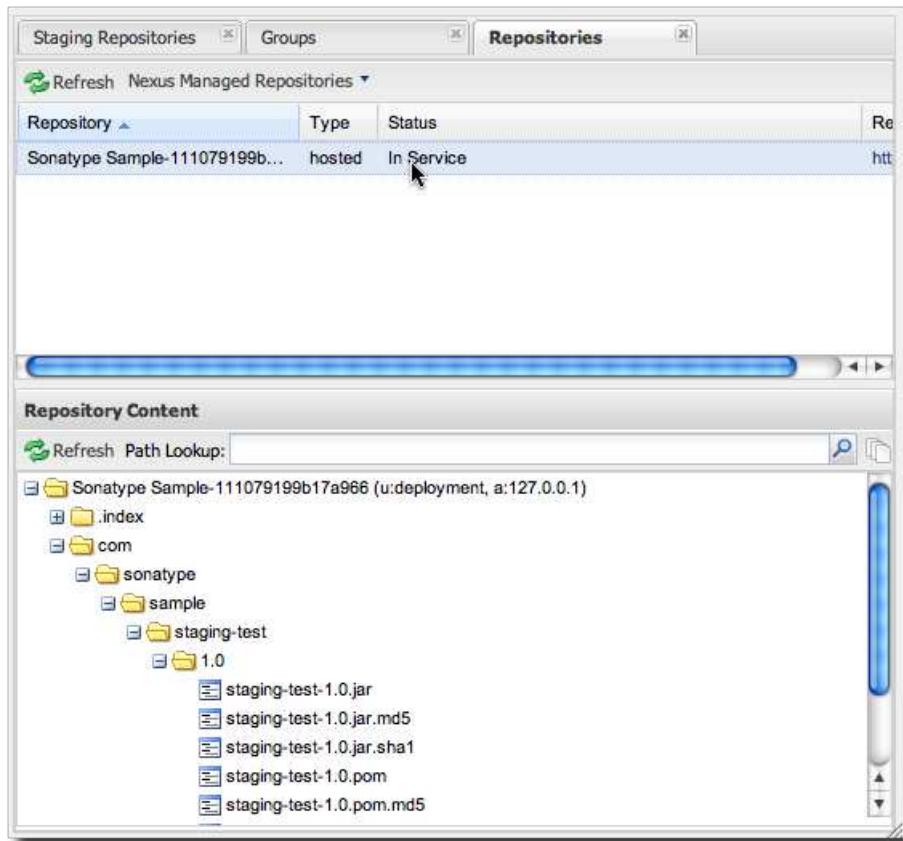


Figure 10.20: Browsing a Staging Repository

You can browse the contents of a staging repository from the Staging Repositories panel. Click on Staging Repositories in the Nexus menu, click on a Staging Repository to browse the contents and perform operations a staging repository.



Figure 10.21: Browsing Repository via Staging Profiles

10.6.3 Releasing a Staging Repository

Once you are finished testing or certifying that the contents of a Staging Repository are correct, you are ready to either Release or Drop the Staging Repository. Dropping the Staging Repository will delete the temporary staging repository from Nexus and remove any reference to this repository from the groups it was associated with. Releasing the Staging Repository allows you to publish the contents of this temporary repository to a Hosted repository.

To release a Staging Repository select Staging from the Nexus menu and then click on the appropriate Staging Profile. This will display a list of Staging Repositories associated with that Staging Profile. To release the contents of a repository, load the list of Staging Repositories, check the box next to the staging repository you wish to promote and then click the Release button shown in Figure 10.22.

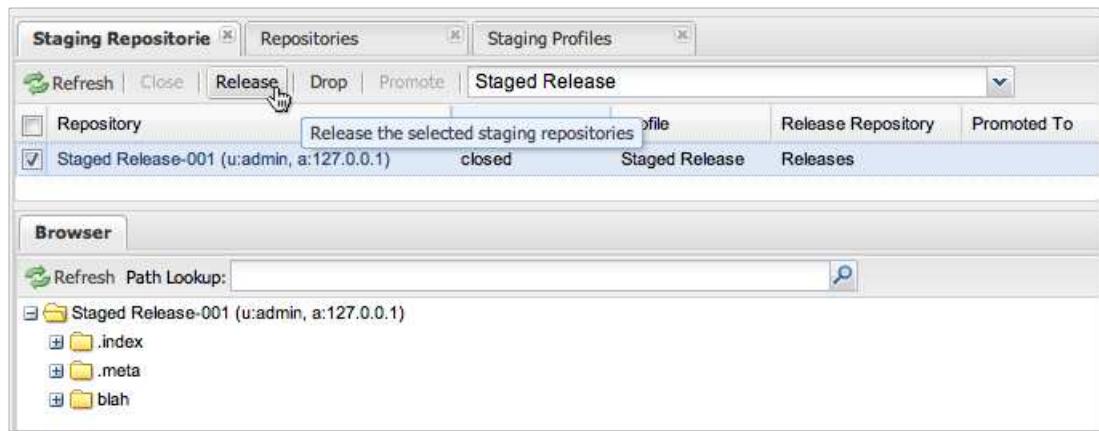


Figure 10.22: Promoting a Staging Repository

Once you click Release, the Nexus Staging Suite will ask you to supply a description for this release action.

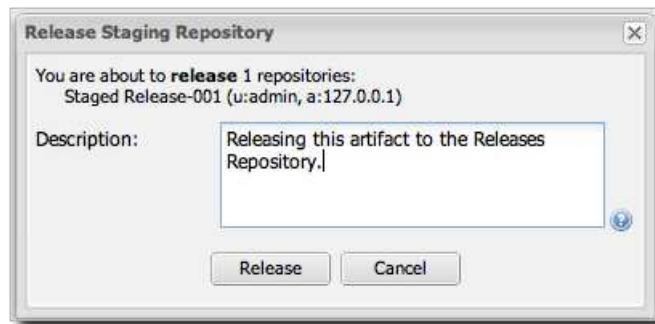


Figure 10.23: Selecting the Destination Repository for Staged Repository Promotion

Supplying a description and clicking on Release will publish the contents of a Staging Repository to a Hosted repository and delete the Staging Repository from Nexus.



Figure 10.24: Confirmation Dialog for Repository Promotion

10.6.4 Promoting a Staging Repository

If you have a staging repository that you want to promote to a Build Promotion profile, open the list of Staging Repositories by selecting Staging Repositories from the Nexus menu, select the repository you intend to promote, and click the Promote button as shown in Figure 10.25.

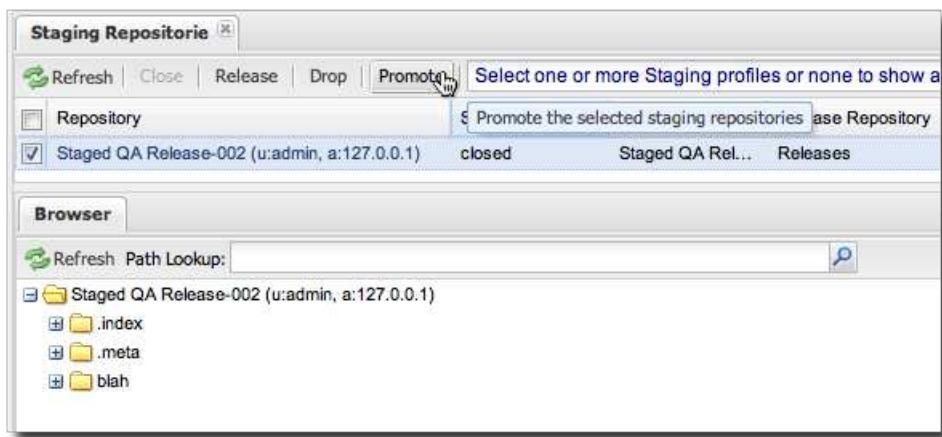


Figure 10.25: Promoting a Staging Repository

After clicking the Promote button the Promote Staging Repository shown in Figure 10.26 will be dis-

played. In this dialog, you can choose the Build Promotion profile to promote the staging repository to, and you can supply a short description of the promotion. Clicking on the Promote button in this dialog will promote the staging profile to a build promotion profile and expose the contents of the selected staging repository through a group associated with the build promotion profile.

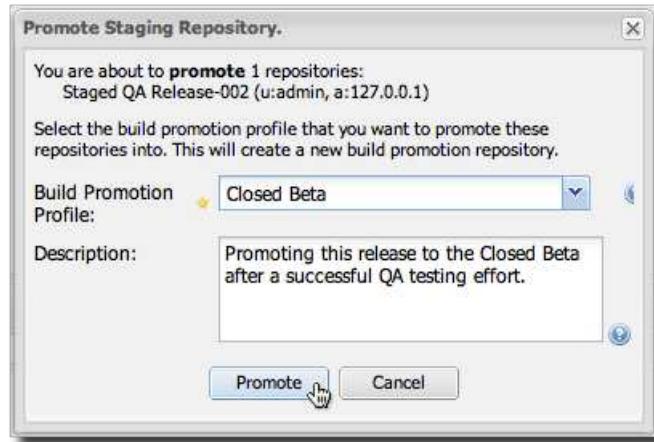


Figure 10.26: Multi-level Staging and Build Promotion

After you promote a staging repository to a Build Promotion profile the build promotion profile will create a temporary repository which contains the contents of the promoted staging repository. The staging repository will be a Group Member of the Build Promotion repository. One or more staging repositories can be promoted to a single Build Promotion profile, and you can browse the Group Member by selecting the Build Promotion repository and viewing the Group Member tab as shown in Figure 10.27.

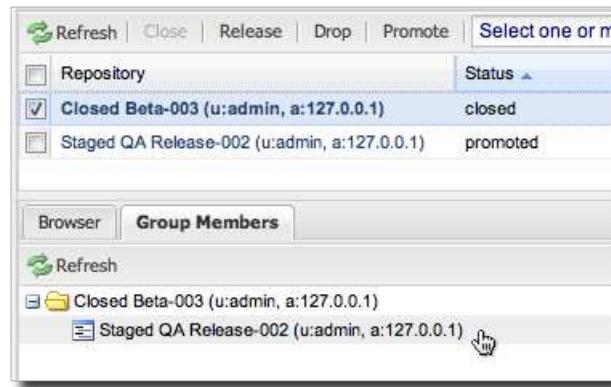


Figure 10.27: Multi-level Staging and Build Promotion

10.6.5 Releasing, Promoting, and Dropping Build Promotion Profiles

When you configure a Build Promotion profile and promote Staging Repositories to promotion profiles, each Build Promotion profile creates a repository which contains one or more Staging Repositories. Just like you can promote the contents of a Staging Repository to a Build Promotion profile, you can also promote the contents of a Build Promotion profile to another Build Promotion profile. When you do this you can create hierarchies of staging repositories and build promotion profiles which can then be dropped or released together.

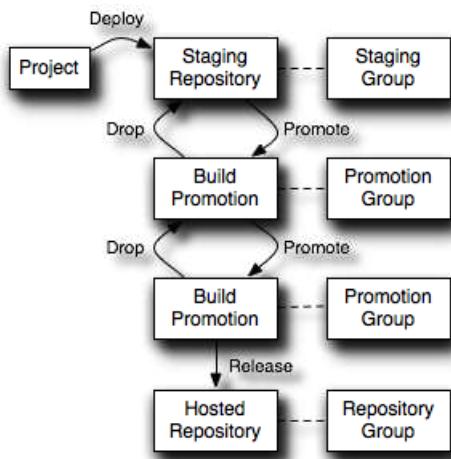


Figure 10.28: Releasing, Promoting, and Dropping Build Promotion Profiles

When you promote a staging repository to a build promotion profile, you make the contents of a staging repository available via a repository group associated with a build promotion profile.

For example, if you staged a few artifacts to a QA Staging Repository and then subsequently promoted that repository to a Closed Beta Build Promotion group, the contents of the QA Staging Repository would initially be made available via a QA Repository Group. After a build promotion, these artifacts would also be available via a Closed Beta repository group.

You can take it one step further and promote the contents of the Closed Beta Build Promotion profile to yet another Build Promotion profile. In this way you can have an arbitrary number of intermediate steps between the initial staging deployment and the final release.

If you drop the contents of a build promotion profile, you roll back to the previous state. For example, if you decided to drop the contents of the Closed Beta build promotion group, Nexus will revert the status of the Staging Repository from promoted to closed, and make the artifacts available via the QA Staging Repository. The effects of promoting, dropping, and releasing artifacts through a series of Staging Profiles and Build Promotion Profiles is shown in Figure 10.28.

When you perform a release on a Build Promotion profile, each Staging Repository is going to release artifacts to the Release Repository configured in Figure 10.8. Because a Build Repository can contain one or more promoted staging repositories, this means that releasing a Build Promotion profile can cause

artifacts to be published to more than one Release Repository. Build Promotion profiles are not directly related to release repositories, only staging profiles are directly associated with target release repositories. Figure 10.29 illustrates this behaviour with two independent Staging Repositories each configured with a separate Release Repository. Releasing the Build Promotion profile causes Nexus to publish each Staging Repository to a separate hosted repository.

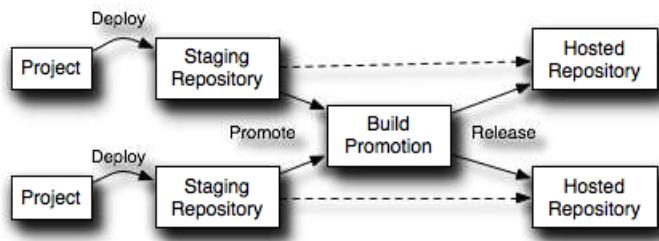


Figure 10.29: Promoting Multiple Repositories to the Same Build Promotion Profile

10.7 Managing Staging Repositories with the Nexus Maven Plugin

You can do everything that was described in Section 10.6 with the Nexus Maven Plugin. Using the Nexus Maven Plugin you can:

- Close a Staging Repository
- Promote a Staging Repository
- Drop a Staging Repository
- List All Available Staging Repositories

10.7.1 Running the Nexus Maven Plugin

To invoke goals in the Nexus Maven plugin, you will want to add the appropriate plugin group to your Maven settings file. Add the org.sonatype.plugins groupId to `~/.m2/settings.xml` as shown in [Adding org.sonatype.plugins to pluginGroups in Maven Settings](#).

Adding org.sonatype.plugins to pluginGroups in Maven Settings

```
<settings>
  ...
  <pluginGroups>
    <pluginGroup>org.sonatype.plugins</pluginGroup>
  </pluginGroups>
  ...
</settings>
```

Adding the org.sonatype.plugins group to your Maven Settings will allow you to run the following goals from the Nexus Maven Plugin:

nexus:staging-close

This goal will close a staging repository from Maven. This goal in the Nexus Maven plugin corresponds to the procedure described in Section [10.6.1](#).

nexus:staging-list

This goal will list all of the staging repositories which are currently visible to a user.

nexus:staging-drop

This goal allows you to drop a specific staging repository. If no repositories are specified for this goal, this plugin will present an interactive menu listing all of the closed staging repositories currently eligible for a drop operation.

nexus:staging-promote

This goal allows you to promote a specific repository. If no repositories are specified for this goal, this plugin will present an interactive menu listing all of the closed staging repositories currently eligible for a promote operation.

Once you have configured the pluginGroup in your Maven Settings file, you can run the Nexus Maven plugin from the command line. In order to access the staging suite in your Nexus instance, the plugin must be told where Nexus is.

```
$ mvn nexus:staging-list
```

10.7.2 Configuring Nexus Maven Plugin for Staging

All of the Staging goals in the Nexus Maven plugin require security credentials and a base URL for the Nexus server you are attempting to manage. You can specify security credentials by supplying a

username and password or by supplying a server id that corresponds to a server in your Maven Settings (~/.m2/settings.xml). The common configuration parameters and security configuration properties are:

nexusUrl

Points to the Nexus server installation's base URL. If you have installed Nexus on your local machine, this would be <http://localhost:8081/nexus/>

username

Username to use for authenticating to Nexus. Default value is operating system level user name

password

Password to use for authenticating to Nexus

serverAuthId

You should specify either username and password or the serverAuthId. If you specify a value for serverAuthId, the Nexus Maven plugin is going to look at the contents of your ~/.m2/settings.xml file and use the username and password from a server definition. In most cases a valid user login will be required to access your staging information. By default, if you don't specify the nexusUrl and password parameters, the plugin will prompt you for them. If you don't specify the username parameter, the Java System property \$user.name. In addition to these security options, all of the staging goals have a common configuration property which controls the logging level.

verboseDebug

If verboseDebug is set to true Maven will print out debug messages that detail the plugin's interaction with Nexus.

10.7.3 Listing Your Open Staging Repositories

Once you've deployed one or more sets of artifacts as release candidate to Nexus, you'll have one or more open staging repositories. There are a variety of actions you can take with these repositories, but maybe one of the most basic is to list them. This gives you a pretty good view into the status of your release(s). The basic command is:

```
$ mvn nexus:staging-list
[...]
[INFO] Logging into Nexus: http://localhost:8082/nexus
[INFO] User: testuser
[INFO]

[INFO] The following OPEN staging repositories were found:
- staging-003 (profile: Example Profile)
URL: http://localhost:8082/nexus/content/repositories/staging-003
```

```
[INFO] The following CLOSED staging repositories were found:  
- staging-001 (profile: Example Profile)  
URL: http://localhost:8082/nexus/content/repositories/staging-001  
Description: This is a test repository  
  
- staging-002 (profile: Example Profile)  
URL: http://localhost:8082/nexus/content/repositories/staging-002  
Description: This is another test repository  
  
You can find more information about this Mojo  
https://repository.sonatype.org/content/sites/maven-sites/nexus-maven- ↵  
plugin/usage-staging.html [here]
```

10.7.4 Closing a Staging Repository

Before your team can run any tests against the set of artifacts that constitute your release, you need to mark the open staging repository as closed. This means that no additional artifacts can be added to that specific staging repository, making the set of artifacts it contains an immutable snapshot. When it is closed, the repository will become available for artifact resolution. The basic command is:

```
$ mvn nexus:staging-close  
[INFO]  
  
Available Staging Repositories:  
  
1: staging-002 (profile: Example Profile)  
URL: http://localhost:8082/nexus/content/repositories/staging-002  
  
Select a repository to close (1) 1: : 1  
  
Repository Description: This is a test repository  
[INFO] Finishing staging repository for: 'com.myco:my-project:1':  
  
- staging-002 (profile: Example Profile)  
URL: http://localhost:8082/nexus/content/repositories/staging-002  
  
[INFO] The following CLOSED staging repositories were found for: \'  
'com.myco:my-project:1':  
  
- staging-001 (profile: Example Profile)  
URL: http://localhost:8082/nexus/content/repositories/staging-001  
Description: This is a test repository
```

```
- staging-002 (profile: Example Profile)
URL: http://localhost:8082/nexus/content/repositories/staging-002
Description: This is another test repository
```

The output above shows that the staging-close goal found an open staging repository - staging-001 - for the current project, then told Nexus to close it. Afterwards, it displayed the list of closed staging repositories, which included the one we just closed. If you don't have an open staging repository, you'll see something like this instead:

```
No open staging repositories found. Nothing to do!

[INFO] The following CLOSED staging repositories were found for: \
'com.myco:my-project:1':

- staging-001 (profile: Example Profile)
URL: http://localhost:8082/nexus/content/repositories/staging-001
Description: This is a test repository

- staging-002 (profile: Example Profile)
URL: http://localhost:8082/nexus/content/repositories/staging-002
Description: This is another test repository

You can find more information about this Mojo https://repository.sonatype. ↵
org/content/sites/maven-sites/nexus-maven-plugin/staging-close-mojo. ↵
html[here]
```

10.7.5 Dropping a Closed Staging Repository

In the unfortunate event that your project artifacts fail during testing, you may need to drop the staging repository that houses them, in order to avoid confusing them with newer candidate releases. The basic command is:

```
$ mvn nexus:staging-drop
[INFO]

Available Staging Repositories:

1: staging-006 (profile: Example Profile)
URL: http://localhost:8082/nexus/content/repositories/staging-006
Description: This is a test repository

Select a repository to drop (1) 1: : 1
[INFO] Dropping staged repository:
```

```
- staging-006 (profile: Example Profile)
URL: http://localhost:8082/nexus/content/repositories/staging-006
Description: This is a test repository
```

The goal will present you with a list of closed staging repositories, with the first in the list selected as the default response. If you simply hit the Enter key, the default will be used; otherwise, the repository corresponding to the number you select will be used. If you have no closed staging repositories, you'll see something like this instead:

```
[INFO]

No closed staging repositories found. Nothing to do!

You can find more information about this Mojo https://repository.sonatype.org/content/sites/maven-sites/nexus-maven-plugin/staging-drop-mojo.html [here]
```

10.7.6 Promoting a Closed Staging Repository

On the other hand, if your project artifacts pass all tests, you will find that you need to promote the staging repository that houses them, in order to finalize the release and make the artifacts available for public consumption. The basic command is:

```
$ mvn nexus:staging-promote
[INFO]

Available Staging Repositories:

1: staging-006 (profile: Example Profile)
URL: http://localhost:8082/nexus/content/repositories/staging-006
Description: This is a test repository

Select a repository to promote (1) 1: : 1
Target Repository ID: releases
[INFO] Promoting staging repository to: releases:

- staging-006 (profile: Example Profile)
URL: http://localhost:8082/nexus/content/repositories/staging-006
Description: This is a test repository
```

The goal will present you with a list of closed staging repositories, with the first in the list selected as the default response. If you simply hit the Enter key, the default will be used; otherwise, the repository

corresponding to the number you select will be used. If you have no closed staging repositories, you'll see something like this instead:

```
[INFO]
```

```
No closed staging repositories found. Nothing to do!
```

```
You can find more information about this Mojo https://repository.sonatype.org/content/sites/maven-sites/nexus-maven-plugin/staging-promote-mojo.html [here]
```

Chapter 11

Repository Health Check

Repository Health Check is a feature of Nexus that integrates data used for [Sonatype Insight](#). Sonatype Insight is a separate product that consists of tools to monitor and manage license, quality and security data about artifacts used in your software development life cycle.

Repository Health Check provides access to a limited subset of the available data in Sonatype Insight via the Sonatype Insight Service right in your Nexus server. The Sonatype Insight Service is part of the Central Repository and exposes data about the artifacts there, including license information, security vulnerability data and other statistics like usage data and download numbers.

To perform a Repository Health Check about the artifacts in your Nexus instance, you have to click the Analyze button in the Quality column of the list of Repositories displayed in Figure 11.1. In order to be able to do that you have to be logged in as an administrator. Once the data gathering analysis is completed the Quality column will display the number of security and license issues found.

Repository	Type	Quality	Format	Policy	Repository Status	Repository Path
Apache Snapshots	proxy	UNAVAILABLE	maven2	Snapshot	In Service	https://repository.sonatype.org/content/repositories/apache-snapshots/
Apache Staging	proxy	ANALYZE	maven2	Release	In Service	https://repository.sonatype.org/content/repositories/apache-staging/
Atlassian	proxy	ANALYZE	maven2	Release	In Service	https://repository.sonatype.org/content/repositories/atlassian-releases/
Atlassian User Contrib	proxy	ANALYZE	maven2	Release	In Service	https://repository.sonatype.org/content/repositories/atlassian-user-contrib-releases/
ben-tests	hosted	UNAVAILABLE	maven-site		In Service	https://repository.sonatype.org/content/repositories/ben-tests/
Central Proxy	proxy	2117 / 19162	maven2	Release	In Service	https://repository.sonatype.org/content/repositories/central-proxy/
CentralM1	virtual	UNAVAILABLE	maven1	Release	In Service	https://repository.sonatype.org/content/repositories/centralm1-releases/
Codehaus Snapshots	proxy	UNAVAILABLE	maven2	Snapshot	In Service	https://repository.sonatype.org/content/repositories/codehaus-snapshots/
Concierge	proxy	ANALYZE	maven2	Release	In Service - Remo...	https://repository.sonatype.org/content/repositories/concierge-releases/
DDSteps	proxy	ANALYZE	maven2	Release	In Service - Remo...	https://repository.sonatype.org/content/repositories/ddsteps-releases/
Docbkx Snapshots	proxy	UNAVAILABLE	maven2	Snapshot	In Service	https://repository.sonatype.org/content/repositories/docbkx-snapshots/

Figure 11.1: The Repositories List with Different Quality Status Indicators and Result Counts

Hovering your mouse pointer of that value will display the Repository Health Check summary data in a pop up window. A sample window is displayed in Figure 11.2.

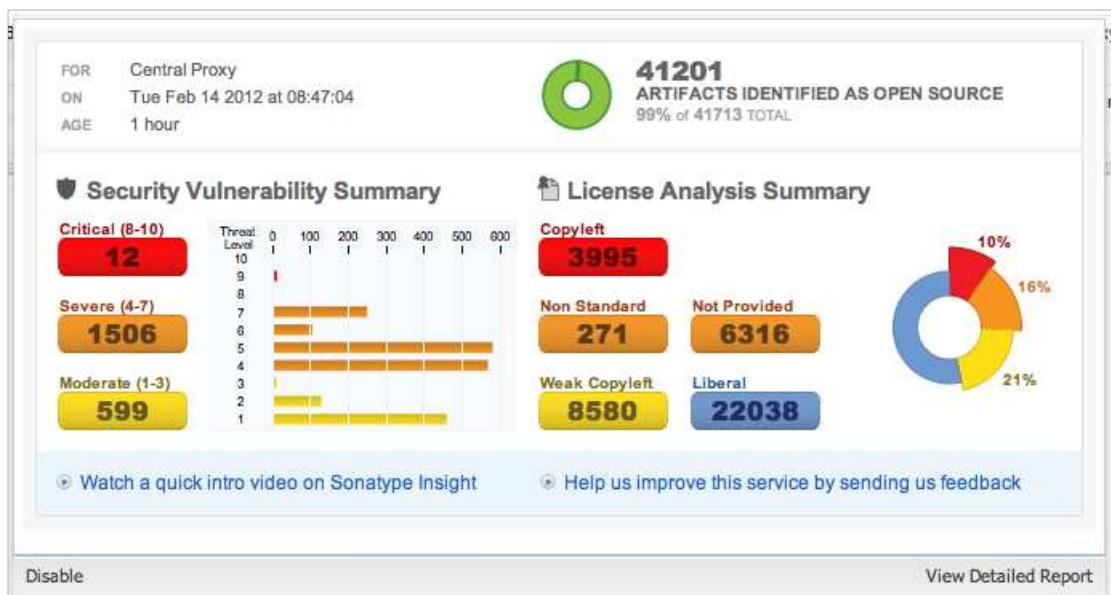


Figure 11.2: A Result Summary Window for a Repository Health Check

At the bottom of the pop up window you find a button to disable the analysis for this repository as well as the button View Detailed Report to access the detailed report. It will show up in another tab in the main area of the Nexus user interface.

The detailed report contains the same overview data and charts for security and license information at the top displayed in Figure 11.3 .

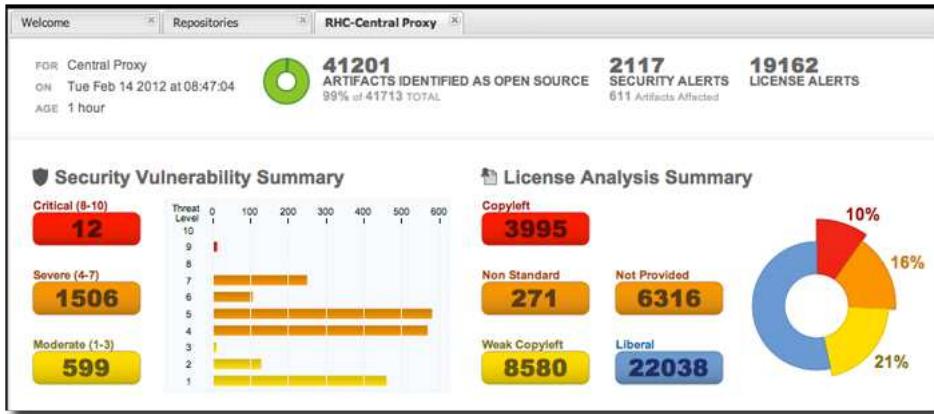


Figure 11.3: Summary of the Detailed Repository Health Check Panel

Below this overview a drop down for Security and License information allows you to toggle between two lists displaying further details. Both lists have a filter for each column at the bottom of the list that allows you to narrow down the number of rows in the table and therefore find specific entries easily.

The security list as visible in Figure 11.4 contains columns for Threat Level, Problem Code and the GAV parameters identifying the affected artifact. The Problem Code column is a link to the security warning referenced and commonly links a specific entry in the [Open Source Vulnerability Database](#) or the [Common Vulnerabilities and Exposures](#) list. Both of these databases have a descriptive text for the vulnerability and further information and reference links.

View By: Vulnerabilities

Threat Level	Problem Code	Group	Artifact	Version
7	CVE-2010-2076	org.apache.cxf	cxf-common-utilities	2.2.4
	CVE-2011-3190	org.apache.tomcat	coyote	6.0.33
	osvdb-24364	struts	struts	1.1-rc1
	osvdb-67294	org.apache.cxf	cxf-common-utilities	2.2
	osvdb-24363	struts	struts	1.1-rc1
	osvdb-67294	org.apache.cxf	cxf-common-utilities	2.2.4
	CVE-2011-3190	org.open.rules	org.open.rules.tomcat.lib	5.7.2
	osvdb-74818	org.ow2.jonas.assemblies.profiles	jonas-full	5.3.0-M2
	CVE-2006-1547	struts	struts	1.1-rc1
	osvdb-67294	org.apache.cxf	cxf-common-utilities	2.1.2
	CVE-2010-2076	org.apache.cxf	cxf-common-utilities	2.2
	CVE-2010-2076	org.apache.cxf	cxf-common-utilities	2.1.2
	osvdb-65697	org.apache.axis2	axis2-webapp	1.5.1
	osvdb-67294	org.apache.cxf	cxf-bundle-jaxrs	2.2
	CVE-2006-1546	struts	struts	1.1-rc1
	CVE-2010-2076	org.apache.cxf	cxf-bundle-jaxrs	2.2
	osvdb-65697	org.apache.ws.commons.axiom	axiom-api	1.2.6
	osvdb-65697	org.apache.servicemix	apache-servicemix	4.3.0
	osvdb-74818	org.open.rules	org.open.rules.tomcat.lib	5.7.2
	osvdb-74818	org.apache.tomcat	coyote	6.0.33

Figure 11.4: The Security Data in the Detailed Repository Health Check Report

The license list as visible in Figure 11.5 shows a derived threat in the Effective License Threat column. The Declared License column details the license information found in pom file. The Observed Licenses in Source columns lists all the licences found in the actual source code of the library in the form of file headers and license files. The next columns for the GAV parameters allow you to identify the artifact. The last column Security Issues displays an indicator for potentially existing security issue for the same artifact.

License Threat	Declared License	Observed Licenses in	Group	Artifact	Version
GPL	Apache-2.0	Apache-2.0, GPL	org.sonatype.configuration	base-configuration	1.1
GPL-2.0+	Apache-2.0+, BSD, EPL-	Apache-2.0, BSD, EPL-1	biz.sourceforge_code	base64coder	2010-12-19
GPL, GPL-2.0	CDDL, GPL, GPL-2.0	Not Provided	org.glassfish.core	glassfish	3.1-b13
GPL, GPL-2.0	CDDL, GPL, GPL-2.0	Not Provided	org.glassfish	jaxws-jms	3.1
GPL-2.0, GPL-2.0+	Apache-2.0	Apache-1.1, Apache-2.0,	org.apache.servicemix	servicemix-scripting	2008.01
GPL, GPL-2.0	CDDL, GPL, GPL-2.0	Not Provided	org.glassfish	jaxws-transaction	10.0-b28
GPL	Apache-2.0	Apache, Apache-2.0, GP	org.apache.camel	camel-jms	2.3.0
GPL	AFL-2.1, Apache-2.0, BS	AFL-2.1, Apache-2.0, BS	org.cometd	cometd-demo	1.1.3
GPL-2.0+	GPL-2.0-with-classpath+	GPL-2.0+	me.springframework	spring-me-sample-j2	1.0
GPL	Apache-2.0	Apache-2.0, GPL	org.apache.camel	camel-core	2.1.0
GPL	GPL	Not Provided	org.glassfish.pfl	pfl-asm	3.2.0-b001
GPL, GPL-2.0	CDDL, GPL, GPL-2.0	Not Provided	com.sun.xml.ws	policy	2.0-b01
GPL	Not Provided	GPL, MIT	org.scala-tools.sxr	sxr_2.7.2	0.2
GPL, GPL-3.0+	GPL	GPL-3.0+	org.sonatype.nexus.resolver	nexus-restlight-m2-s	1.8.0
GPL, GPL-2.0	Apache-2.0	Apache-2.0, CDDL-1.0, C	org.apache.servicemix.i	org.apache.servicemix.i	2.1.13_1
GPL-3.0	Apache-2.0	Apache-2.0, GPL-3.0	org.ujoframework	ujof-core	1.20
GPL, GPL-2.0	CDDL, GPL, GPL-2.0	Not Provided	com.sun.grizzly	grizzly-websockets	1.9.36
GPL, GPL-2.0	CDDL, GPL, GPL-2.0	Not Provided	org.glassfish.web	web-gui-plugin-common	3.1.1

Figure 11.5: The License Data in the Detailed Repository Health Check Report

Nexus will report all artifacts in the local storage of the respective repository in the detail panel. This means that at some stage a build running against your Nexus instance required these artifacts and caused Nexus to download them to local storage.

To determine which project and build caused this download to be able to fix the offending dependency by upgrading to a newer version or removing it with an alternative solution with a more suitable license you will have to investigate all your project.

Sonatype Insight itself helps with these tasks by enabling monitoring of builds and products, analyzing release artifacts and creating bill of material and other reports.

Chapter 12

Managing Maven Settings

12.1 Introduction

Nexus Professional allows you to define templates for Maven Settings and use Nexus as a way to distribute updates to a global Maven Settings template. When you move an organization to a repository manager such as Nexus, one of the constant challenges is keeping everyone's Maven Settings synchronized.

If a Nexus administrator makes a change which requires every developer to modify his or her `~/.m2/settings.xml` file, this feature can be used to manage the distribution of Maven Settings changes to the entire organization. Once you have defined a Maven Settings template in Nexus Professional, developers can then use the Nexus Maven plugin to retrieve the new Maven Settings file directly from Nexus Professional.

12.2 Manage Maven Settings Templates

To manage Maven Settings templates, click on Maven Settings in the Enterprise section of the Nexus menu on the left side of the Nexus UI. Clicking on Maven Settings will load the panel shown in Figure 12.1.

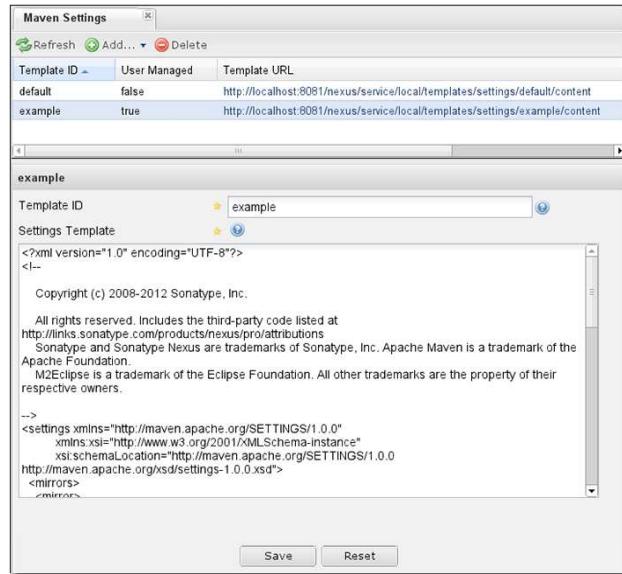


Figure 12.1: The Maven Settings Panel

The Maven Settings panel allows you to add, delete, and edit a Maven Settings template. The default template has an id of "default" and is read-only, it contains the recommended settings for a standard Nexus installation. To create a new Maven Settings template, click on the Add... button and select "Settings Template". Once the new template is created, assign a name to the template in the Template ID text input and click the Save button.

To edit a template, click on a template that has a User Managed value of true, and edit the template in the Settings Templatel panel. Once you are finished editing the template, click Save to save the template. When editing the template, the following variables are available:

baseurl

This variable will be replaced with the base URL of the Nexus installation.

userId

This variable will be replaced with the user id of the user that is generating a Maven Settings file from this template.

To preview a Maven Settings template, click on the Template URL in the table of Maven Settings templates. Clicking on this URL will load a dialog window which contains the Maven Settings file generated

from this template. This rendered view of the Maven Settings template will have all variable references replaced using the current context of the user: \$baseurl will be replaced with the Base URL of the current Nexus installation, and \$userId will be replaced with the user id of the current user.

12.3 Downloading Maven Settings with the Nexus Maven Plugin

Once you have defined a set of Maven templates, you can use the Nexus Maven plugin to distribute changes to the Maven Settings file to the entire organization.

12.3.1 Running the Nexus Maven Plugin

To invoke goals in the Nexus Maven plugin, you will initially have to use a fully qualified groupId and artifactId as shown in [Invoking the Nexus Maven Plugin goal settings-download with a fully qualified groupId and artifactId](#).

Invoking the Nexus Maven Plugin goal settings-download with a fully qualified groupId and artifactId

```
mvn org.sonatype.plugins:nexus-maven-plugin:settings-download
```

To be able to invoke the plugin with the simple identifier "nexus" as shown in [Invoking the Nexus Maven Plugin goal settings-download with a short identifier](#), you have to add the appropriate plugin group to your Maven Settings file as shown in [Adding org.sonatype.plugins to pluginGroups in Maven Settings](#). An initial invocation of the settings-download goal will update your settings file, with a template from Nexus Professional. Every default template in Nexus Professional adds the org.sonatype.plugins group to the pluginGroups, so you will not have to do this manually. It is essential that you make sure that any new, custom templates also include this plugin group definition. Otherwise, there is a chance that a developer could update his or her Maven Settings and lose the ability to use the Nexus Maven plugin with the short identifier.

Invoking the Nexus Maven Plugin goal settings-download with a short identifier

```
mvn nexus:settings-download
```

Adding org.sonatype.plugins to pluginGroups in Maven Settings

```
<settings>
```

```
...
<pluginGroups>
  <pluginGroup>org.sonatype.plugins</pluginGroup>
</pluginGroups>
...
</settings>
```

The "settings-download" goal of the Nexus Maven Plugin downloads a Maven Settings file from Nexus Professional and stores it on a local machine. This goal can be configured to update a user's `~/.m2/settings.xml` file or the global configuration file which is stored in the Maven installation directory. If you are replacing a Maven Settings file, this goal can be configured to make a backup of an existing Maven Settings file.

12.3.2 Configuring Nexus Maven Plugin for Settings Management

The Settings Management goal in the Nexus Maven plugin requires security credentials and a base URL for the Nexus server you are interacting with. You can specify security credentials by supplying a username and password or by supplying a server id that corresponds to a server in your Maven Settings (`~/.m2/settings.xml`). The common configuration parameters and security configuration properties are:

nexusURL

Points to the Nexus server installation's base URL. If you have installed Nexus on your local machine, this would be `http://localhost:8081/nexus/`

username

Username to use for authenticating to Nexus. Default value is `$user.name`.

password

Password to use for authenticating to Nexus

serverAuthId

You should specify either username and password or the serverAuthId. If you specify a value for serverAuthId, the Nexus Maven plugin is going to look at the contents of your `~/.m2/settings.xml` file and use the username and password from a server definition.

In most cases a valid user login will be required to access your settings templates. By default, if you don't specify the `nexusURL` and `password` parameters, the plugin will prompt you for them. If you don't specify the `username` parameter, the Java System property `$user.name` will be used.

In addition to these security options, all of the Maven Settings management goals have the following configuration parameters:

verboseDebug

If verboseDebug is set to true Maven will print out debug messages that detail the plugin's interaction with Nexus.

backupFormat

When backing up an existing settings.xml file, use this date format in conjunction with SimpleDateFormat to construct a new filename of the form: settings.xml.\$(format). Date stamps are used for backup copies of the settings.xml to avoid overwriting previously backed up settings files. This protects against the case where the download mojo is used multiple times with incorrect settings, where using a single static backup-file name would destroy the original, pre-existing settings. Default value is: yyyyMMdd_HHmmss.

destination

The standard destination where the downloaded settings.xml template should be saved. If the destination is "global", the Nexus Maven plugin will save the Maven Settings file to \$M2_HOME/conf. Is the destination is "user", the Nexus Maven plugin will save the Maven Settings file to ~/.m2/settings.xml. If the target parameter is set, it will override this value. Default value is: user.

doBackup

If true and there is a pre-existing settings.xml file in the way of this download, backup the file to a date-stamped filename, where the specific format of the date-stamp is given by the backupFormat parameter. Default value is: true.

encoding

Use this parameter to define a non-default encoding for the settings file.

target

If set, ignore the standard location given by the destination parameter, and use this file location to save the settings template instead. If this file exists, it will be backed up using the same logic as the standard locations (using the doBackup and backupFormat parameters).

url

The full URL of a settings template available from a particular Nexus Professional instance. If missing, the mojo will prompt for this value.

12.3.3 Downloading Maven Settings

To download Maven Settings from Nexus Professional, you will need to know the URL of the Maven Settings template. If you omit this URL on the command-line, the Nexus Maven plugin will prompt you for a URL when it is executed:

```
$ mvn org.sonatype.plugins:nexus-maven-plugin:settings-download
[INFO] [nexus:settings-download]
Settings Template URL: \
.../nexus/service/local/templates/settings/default/content
```

```
[INFO] Existing settings backed up to: \  
/Users/tobrien/.m2/settings.xml.20090408_204422  
[INFO] Settings saved to: /Users/tobrien/.m2/settings.xml
```

Alternatively, you can specify the username, password, and URL on the command line.

```
$ export NX_URL="http://localhost:8081/nexus/"  
$ mvn org.sonatype.plugins:nexus-maven-plugin:settings-download \  
-Durl=${NX_URL}/service/local/templates/settings/default/content \  
-Dusername=admin \  
-Dpassword=admin123</screen>
```

Chapter 13

OSGi Bundle Repositories

13.1 Introduction

Nexus Professional supports the OSGi Bundle Repository format. The OSGi Bundle format is defined by the [OSGi RFC 112 "Bundle Repository"](#). It is a format for the distribution of OSGi "bundles" which includes any components that are described by the OSGi standards set forth in RFC 112. An OBR repository has a single XML file which completely describes the contents of the entire repository. Nexus Professional can read this OBR repository XML and create proxy repositories which can download OSGi bundles from remote OBR repositories. Nexus Professional can also act as a hosting platform for OSGi bundles, you can configure your builds to publish OSGi bundles to Nexus Professional, and then you can expose these bundle repositories to internal or external developers using Nexus Professional as a publishing and distribution platform.

Nexus Professional can also act as a bridge between Maven repositories and OSGi bundle repositories. When you configure a virtual OBR repository which uses a Maven 2 repository as a source repository, Nexus Professional will expose artifacts with the appropriate metadata from the Maven repository as OSGi bundles. In this way, you can unify your OSGi and non-OSGi development efforts and publish artifacts with the appropriate OSGi metadata to Nexus Professional. Non-OSGi clients can retrieve software artifacts from a Maven repository, and OSGi-aware clients can retrieve OSGi bundles from a virtual OBR repository.

The following sections detail the procedures for creating and managing OBR repositories.

13.2 Proxy OSGi Bundle Repositories

Nexus can proxy an OSGi Bundle Repository, using the OBR repository XML as the remote storage location. To create a new proxy OBR repository:

1. Login as an Administrator.
2. Click Repositories in the Left Navigation Menu.
3. Click the Add.. button above the list of Nexus repositories, and choose Proxy repository from the drop-down of repository types.
4. In the New Proxy Repository window,
 - a. Select OBR as the Provider.
 - b. Supply an id and a repository name.
 - c. Enter the URL to the remote repository OBR XML as the Remote Storage location.
 - d. Click Save.

Figure 13.1 provides some sample configuration used to create a proxy of the Apache Felix OBR repository.



Figure 13.1: Creating an OSGi Bundle Proxy Repository

To verify that the OBR proxy repository has been properly configured, you can then load the OBR XML from Nexus Professional. If Nexus Professional is properly configured, you will be able load the obr.xml by navigating to the obr.xml directory:

```
$curl http://localhost:8081/nexus/content/repositories/felix-proxy/.meta/ ←
  obr.xml
<?xml version='1.0' encoding='utf-8'?>
<?xmlstylesheet type='text/xsl' href='http://www.osgi.org/www/obr2html. ←
  xsl'?>
<repository name='Felix OBR Repository' lastmodified='1247493075615'>
  <resource id='org.apache.felix(javax.servlet/1.0.0'
    presentationname='Servlet 2.1 API'
    symbolicname='org.apache.felix(javax.servlet'
    uri='..../bundles/org.apache.felix(javax.servlet-1.0.0.jar'
    version='1.0.0'>
    <description>
      Servlet 2.1 API
    </description>
    <documentation>
      http://www.apache.org/
    </documentation>
    <license>
```

```
http://www.apache.org/licenses/LICENSE-2.0.txt
</license>
...
...
```

13.3 Hosted OSGi Bundle Repositories

Nexus can host an OSGi Bundle Repository, providing you with a way to publish your own OBR bundles. To create an OBR hosted repository:

1. Login as an Administrator.
2. Click Repositories in the Left Navigation Menu.
3. Click the Add.. button above the list of Nexus repositories, and choose Hosted repository from the drop-down of repository types.
4. In the New Hosted Repository window,
 - a. Select OBR as the Provider.
5. Supply an id and a repository name.
6. Click Save.

Figure 13.2 provides some sample configuration used to create a hosted OBR repository.



Figure 13.2: Creating a Hosted OSGi Bundle Repository

13.4 Virtual OSGi Bundle Repositories

Nexus Professional can also be configured to convert a traditional Maven repository into an OSGi Bundle repository using a virtual OBR repository. To configure a virtual OBR repository:

1. Login as an Administrator.
2. Click Repositories in the Left Navigation Menu.
3. Click the Add.. button above the list of Nexus repositories, and choose Virtual repository from the drop-down of repository types.
4. In the New Virtual Repository window,
 - a. Select OBR as the Provider.
 - b. Select another repository's ID in the Source Nexus Repository ID drop-down
 - c. Supply an id and a repository name.
 - d. Click Save.

The next figure provides some sample configuration used to create a virtual OBR repository which transforms the proxy repository for Maven Central into an OBR repository.



Figure 13.3: Creating a Virtual OSGi Bundle Repository from a Maven Repository

13.5 Grouping OSGi Bundle Repositories

Just like Nexus can group Maven repositories, Eclipse update sites, and P2 repositories, Nexus can also be configured to group OSGi Bundle Repositories. To group OSGi bundle repositories:

1. Login as an Administrator.
2. Click Repositories in the Left Navigation Menu.
3. Click the Add.. button above the list of Nexus repositories, and choose Repository Group from the drop-down of repository types.
4. In the New Repository Group window,
 - a. Select OBR Group as the Provider.
 - b. Drag and drop one or more hosted, proxy, or virtual OSGi Bundle repositories into the new group.
 - c. Supply an id and a repository name.
 - d. Click Save.

Figure 13.4 shows an example of the a new repository group which contains a hosted OSGi Bundle repository, a virtual OSGi Bundle repository, and a OSGi Bundle proxy repository.

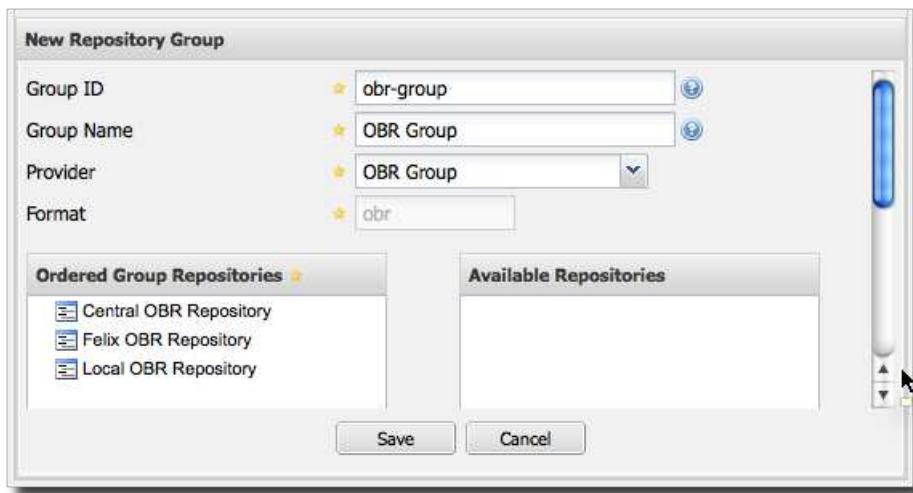


Figure 13.4: Creating a new OSGi Bundle Repository Group

Chapter 14

P2 Repositories

14.1 Introduction

Nexus Professional supports the P2 Repository format. The P2 repository format is a provisioning platform for Eclipse components. For more information about the P2 repository format, see the [Equinox P2 documentation](#) on the Eclipse Wiki.

The following sections detail the procedures for creating and managing P2 repositories.

14.2 Proxy P2 Repositories

Nexus can proxy a P2 Repository. To create a new proxy P2 repository:

1. Login as an Administrator.
2. Click **Repositories** in the Left Navigation Menu.
3. Click the **Add..** button above the list of Nexus repositories, and choose **Proxy repository** from the drop-down of repository types.
4. In the New Proxy Repository window,

- a. Select P2 as the Provider.
- b. Supply an id and a repository name.
- c. Enter the URL to the remote P2 repository as the Remote Storage location.
- d. Click Save.

Figure 14.1 provides some sample configuration used to create a proxy of the Indigo Simultaneous Release P2 repository.

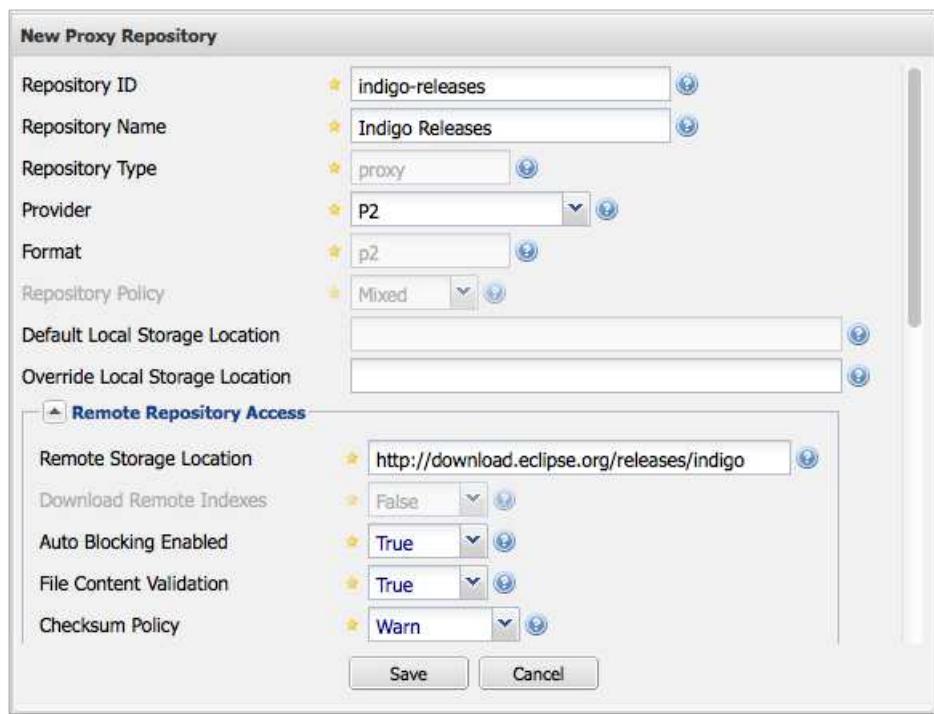


Figure 14.1: Creating a P2 Proxy Repository

14.3 Grouping P2 Repositories

Just like Nexus can group Maven repositories and OBR repositories, Nexus can also be configured to group P2 Repositories. To group P2 repositories:

1. Login as an Administrator.
2. Click Repositories in the Left Navigation Menu.
3. Click the Add.. button above the list of Nexus repositories, and choose Repository Group from the drop-down of repository types.
4. In the New Repository Group window,
 - a. Select P2 as the Provider.
 - b. Drag and drop one or more P2 repositories into the new group.
 - c. Supply an id and a group name.
 - d. Click Save.

Figure 14.2 shows an example of a repository group which contains two P2 proxy repositories.

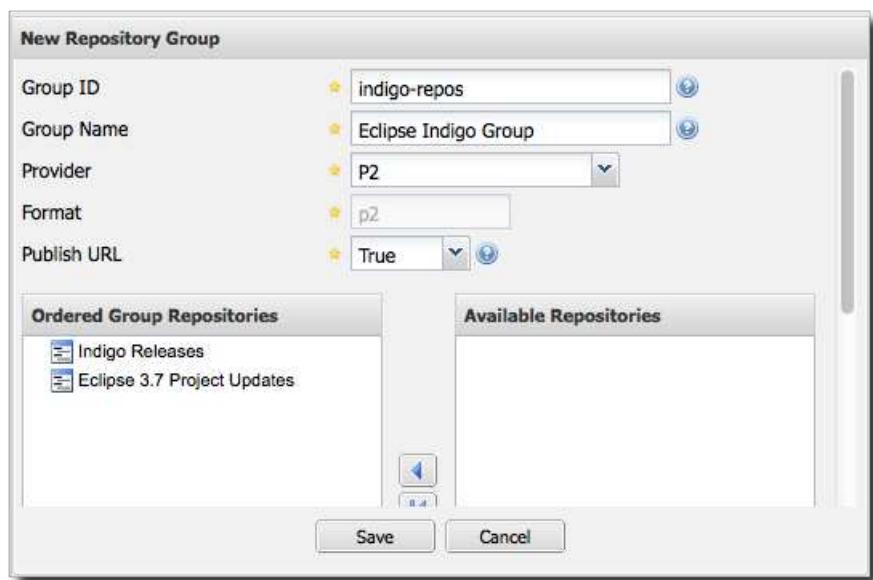


Figure 14.2: Creating a new P2 Repository Group

Chapter 15

.NET Package Repositories

15.1 Introduction

With the recent creation of the [NuGet](#) project a package management solution for .NET developers has become available. Similar to 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 websites at [nuget.org](#) and [nuget.codeplex.com](#) host tool downloads, 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 supports the NuGet repository format for hosted and proxy repositories. Nexus also supports aggregation of NuGet repositories and conversion of other repositories containing ".nupkg" artifacts to the NuGet format. This allows you to improve collaboration and control while speeding up .NET development facilitating open source libraries and sharing of internal artifacts across teams. When you standardize on a single repository for all your development and use it for internal artifacts as well you will get all the benefits of Nexus when working in the .NET architecture.

To share a library or tool with NuGet you create a NuGet package and store it in the Nexus based NuGet repository. Similarly you can use packages others have created and made available in their NuGet repository by proxying it or downloading the package and installing it in your own hosted repository for third party packages.

The NuGet Visual Studio extension allows you to download the package from the repository and install it in your Visual Studio project or solution. NuGet copies everything and makes any required changes to your project setup and configuration files. Removing a package will clean up any changes as required.

15.2 NuGet Proxy Repositories

To proxy an external NuGet repository you simply create a new Proxy Repository as documented in Section 6.2. The Provider has to be set to NuGet. The Remote Storage Location has to be set to the source URL of the repository you want to proxy.

A complete configuration for proxying nuget.org is visible in Figure 15.1.

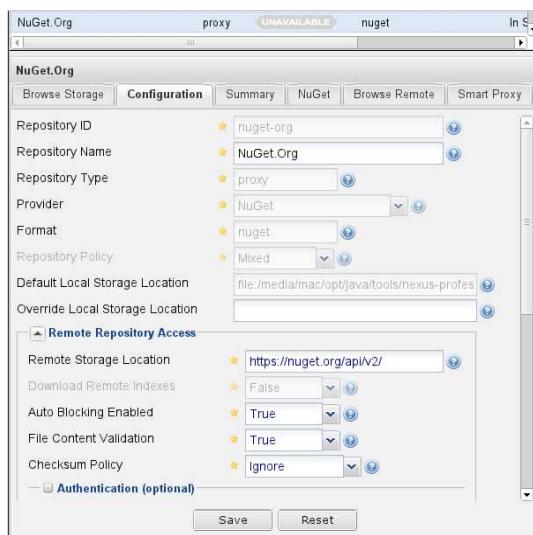


Figure 15.1: NuGet Proxy Repository Configuration for nuget.org

The source URL for the main NuGet.org repository is

```
http://nuget.org/api/v2/
```

The repository configuration for a NuGet proxy repository has an additional tab titled NuGet as visible in Figure 15.2, which displays the Nexus URL at which the repository is available as a NuGet repository.

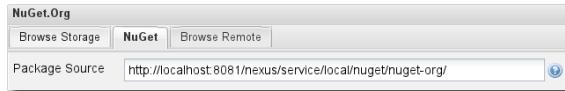


Figure 15.2: NuGet Gallery with Package Source URL

When creating a NuGet proxy repository a Scheduled Task is automatically created to download the index data about the content of the remote NuGet repository. This task is available in the Scheduled Tasks administration section and by default created for a daily schedule. To modify the task access it via the Administration panel in the left hand navigation area and the Scheduled Tasks menu item. The task will be using the name of the proxy repository with (NuGet Feed) appended. A user interface as displayed in Figure 15.3 will allow you to adjust the task as desired.

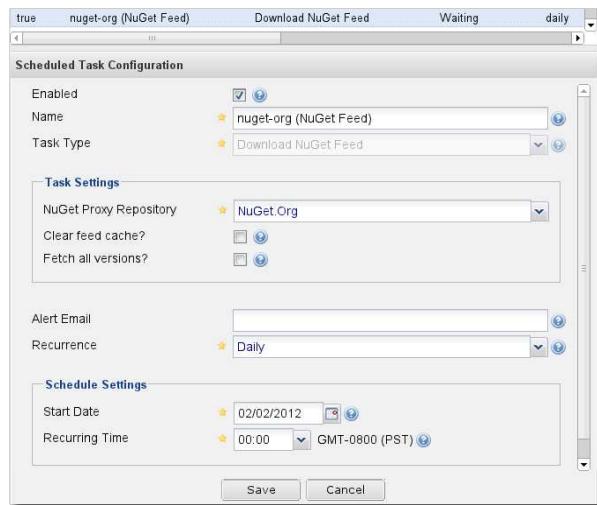


Figure 15.3: NuGet Proxy Repository Scheduled Task

The default task incrementally downloads information about the latest version of published packages. If you want details about all versions you can select the Fetch All Versions checkbox, save the updated configuration and manually trigger the Scheduled Task.

Deleting the proxy repository will remove the scheduled task.

15.3 NuGet Hosted Repositories

A hosted repository for NuGet can be used to upload your own packages as well as third party packages. It is good practice to create two separate hosted repositories for these purposes.

To create a NuGet hosted repository simply create a new Hosted Repository and set the Provider to NuGet. A sample configuration for an internal releases NuGet hosted repository is displayed in Figure 15.4.

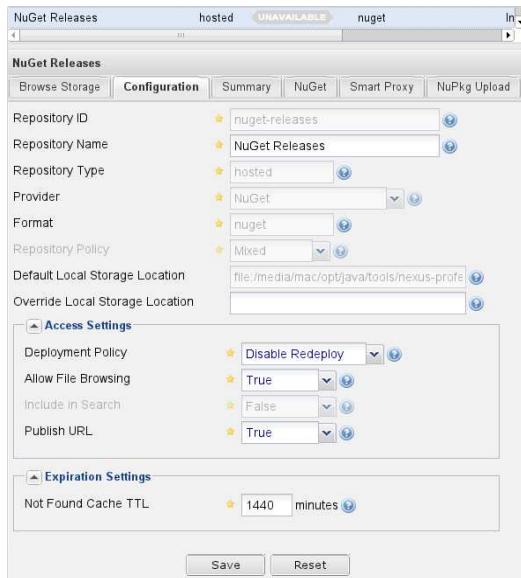


Figure 15.4: Example Configuration for a NuGet Hosted Repository for Release Packages

Besides the NuGet tab the configuration for the repository has a NuPkg Upload tab as displayed in Figure 15.5, that allows you to manually upload one or multiple packages.

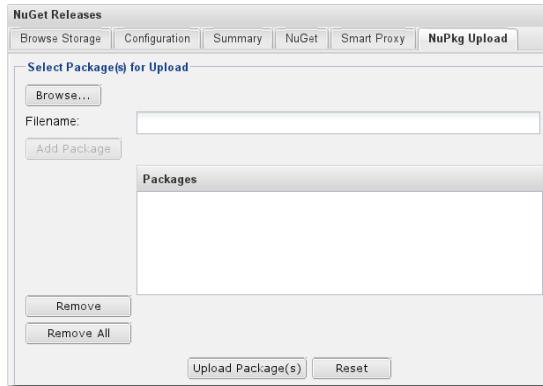


Figure 15.5: The NuPkg Upload Panel for a Hosted NuGet Repository

The NuGet feed is immediately updated as packages are deployed or deleted from the host repository. If for some reason you ever need to rebuild the feed for a hosted NuGet repository you can manually schedule a Rebuild NuGet Feed task.

15.4 NuGet Virtual Repositories

If you have deployed NuGet packages to a Maven repository in the past you can expose them to Visual Studio by creating a Virtual repository and setting the Format to NuGet Shadow Repository. The setup displayed in Figure 15.6 shows a virtual repository set up to expose the content of the regular Maven Releases repository in the form of a NuGet repository, so that NuGet can access any NuGet packages deployed to the releases repository.

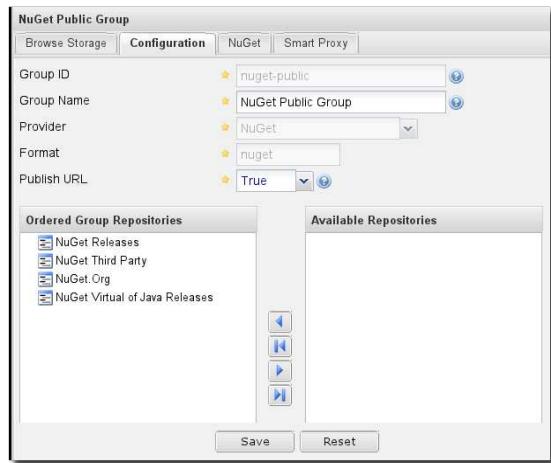


Figure 15.6: A Virtual NuGet Repository for the Maven Releases Repository

The NuGet feed is immediately updated as packages are deployed or deleted from the shadowed repository. If for some reason you ever need to rebuild the feed for a virtual NuGet repository you can manually schedule a Synchronize Shadow Repository task.

15.5 NuGet Group Repositories

A repository group allows you to expose the aggregated content of multiple proxy and hosted repositories with one URL to your tools. This is possible for NuGet repositories by creating a new Repository Group with the Format set to NuGet.

A typical useful example would be to group the proxy repository that proxies nuget.org, an internal releases NuGet hosted repository and a third party Nuget hosted repository. The configuration for such a setup is displayed in Figure 15.7.

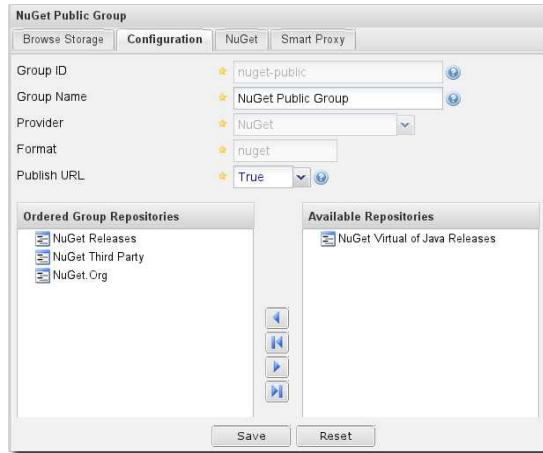


Figure 15.7: A Public Nuget Group Combining a Proxy and Two Hosted Repositories

Using the Repository Path of the repository group as your NuGet repository URL in your client tool will give you access to the packages in all three repositories with one URL.

15.6 Accessing Packages in Repositories and Groups

Once you have set up your hosted and proxy repositories for NuGet packages and potentially created a group you can access them with the nuget tool on the command line. Copy the Package Source url from the NuGet tab of the repository/group configuration you want to access and add it to nuget on the command line with e.g.:

```
nuget sources add -name NuGetNexus -source http://localhost:8081/nexus/ ↵
  service/local/nuget/nuget-public
```

Replace localhost with the public hostname or url of your Nexus server and nuget-public with the name of the repository you want to proxy. Ideally this will be your NuGet group.

After this source was added you can list the available packages with

```
nuget list
```

15.7 Deploying Packages to NuGet Hosted Repositories

In order to authenticate a client against a NuGet repository NuGet uses an API key for deployment requests. These keys are generated separately on-request from a user account on the NuGet gallery and can be re-generated at any time. At regeneration all previous keys generated for that user are invalid.

15.7.1 Creating a NuGet API-Key

For usage with Nexus, API keys are only needed when packages are going to be deployed. Therefore API key generation is by default not exposed in the user interface to normal users. Only users with the Deployer role have access to the API keys.

Other users that should be able to access and create an API key have to be given the Nexus API-Key Access role in the Users Security administration user interface.

In addition the NuGet API-Key Realm has to be activated. To do this, simply add the realm to the selected realms in the Security Settings section of the Server Administration.

Once this is set up you can view, as well as reset, the current Personal API Key in the NuGet tab of any NuGet proxy or hosted repository as visible in Figure 15.8

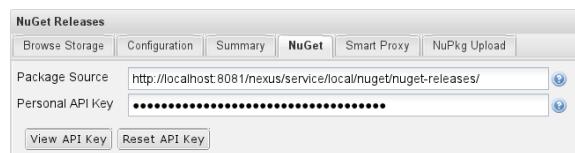


Figure 15.8: Viewing and Resetting the NuGet API Key in the NuGet Configuration Tab

15.7.2 Creating a Package for Deployment

Creating a package for deployment can be done with the pack command of the nuget command line tool or within Visual Studio. Detailed documentation can be found on the [NuGet website](#).

15.7.3 Deployment with the NuPkg Upload User Interface

Manual upload of one or multiple packages is done on the NuPkg Upload tab of the repository displayed in Figure 15.5. Press the Browse button to access the package you want to upload on the file system and press Add Package. Repeat this process for all the packages you want upload and press Upload Package(s) to complete the upload.

15.7.4 Command line based Deployment to a Nexus 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 API Key and the Package Source path. Both of them are available in the NuGet tab of the hosted NuGet repository you want to deploy to. 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 [on-line help](#).

15.8 Integration of Nexus NuGet Repositories in Visual Studio

In order to access a Nexus NuGet repository or preferable all Nexus NuGet repositories exposed in a group you provide the Repository Path in the Visual Studio configuration for the Package Sources of the Package Manager as displayed in Figure 15.9.

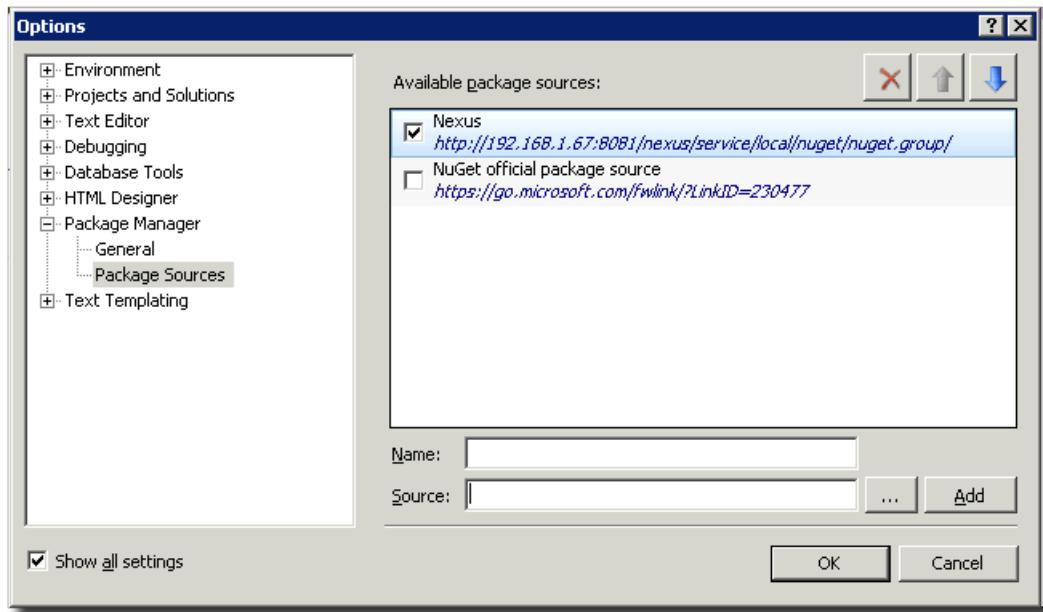


Figure 15.9: Package Source Configuration for the Package Manager in Visual Studio to Access A Nexus NuGet Repository Group

With this configuration in place all packages available in your Nexus NuGet repository will be available in the Package Manager in Visual Studio ready for install.

Chapter 16

Deploying Sites to Nexus

16.1 Introduction

Nexus Professional provides a new hosted repository type: a Maven Site repository. This repository can be used to hold a Maven-generated web site. This chapter details the process of configuring a Maven Site repository and configuring a simple Maven project to publish a Maven-generated project site to an instance of Nexus Professional.

16.2 Creating a New Maven Project

In this chapter, you will be creating a simple Maven project with a simple web site that will be published to a Nexus Professional Maven Site repository. To create a new Maven project, use the archetype plugin's archetype:generate goal on the command line, and supply the following identifiers:

- groupId: org.sonatype.books.nexus
 - artifactId: sample-site
 - version: 1.0-SNAPSHOT
 - package: org.sonatype.books.nexus
-

```
~/examples$ mvn archetype:generate
[INFO] [archetype:generate {execution: default-cli}]
[INFO] Generating project in Interactive mode
Choose archetype:
1: internal -> appfuse-basic-jsf
...
13: internal -> maven-archetype-portlet (A simple portlet application)
14: internal -> maven-archetype-profiles ()
15: internal -> maven-archetype-quickstart ()

...
Choose a number: (...14/15/16...) 15: : 15
Define value for groupId: : org.sonatype.books.nexus
Define value for artifactId: : sample-site
Define value for version: 1.0-SNAPSHOT: : 1.0-SNAPSHOT
Define value for package: org.sonatype.books.nexus: : org.sonatype.books. ↵
    nexus
Confirm properties configuration:
groupId: org.sonatype.books.nexus
artifactId: sample-site
version: 1.0-SNAPSHOT
package: org.sonatype.books.nexus
Y: :
[INFO] Parameter: groupId, Value: org.sonatype.books.nexus
[INFO] Parameter: packageName, Value: org.sonatype.books.nexus
[INFO] Parameter: package, Value: org.sonatype.books.nexus
[INFO] Parameter: artifactId, Value: sample-site
[INFO] Parameter: basedir, Value: /private/tmp
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] OldArchetype created in dir: /private/tmp/sample-site
[INFO] ↵
----- ↵
[INFO] BUILD SUCCESSFUL
[INFO] ↵
----- ↵
[INFO] Total time: 23 seconds
[INFO] Finished at: Sat Oct 03 07:09:49 CDT 2009
[INFO] Final Memory: 13M/80M
[INFO] ↵
----- ↵
```

After running the archetype:generate command you will have a new project in a sample-site/ sub-directory.

16.3 Configuring Maven for Site Deployment

To deploy a site to a Nexus Professional Maven Site repository, you will need to configure the project's distribution management settings, add site deployment information, and then update your Maven settings to include the appropriate credentials for Nexus.

```
~/examples$ cd sample-site
~/examples/sample-site$ more pom.xml
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.sonatype.books.nexus</groupId>
  <artifactId>sample-site</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>sample-site</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Add the following section to sample-site/pom.xml after the dependencies element. This section will tell Maven where to publish the Maven-generated project web site:

Distribution Management for Site Deployment to Nexus

```
<distributionManagement>
  <site>
    <id>nexus-site</id>
    <url>dav:http://localhost:8081/nexus/content/sites/site/</url>
  </site>
</distributionManagement>
```

Note

In [Distribution Management for Site Deployment to Nexus](#), there is a Nexus Professional installation running on localhost port 8081. In your environment you will need to customize this URL to point to your own Nexus instance.

In addition to the distributionManagement element, you will want to add the following build element that will configure Maven to use version 2.0.1 of the Maven Site plugin.

Configuring the Maven Site Plugin for Nexus Site Deployment

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-site-plugin</artifactId>
      <version>2.0.1</version>
    </plugin>
  </plugins>
</build>
```

Note

Site deployment to Nexus Professional requires version 2.0.1 or higher of the Maven Site Plugin.

16.4 Adding Credentials to Your Maven Settings

When the Maven Site plugin deploys a site to Nexus, it will need to supply the appropriate deployment credentials to Nexus. To configure this, you will need to add credentials to your Maven Settings. Open up your `~/.m2/settings.xml` and add the following server configuration to the servers element. If you already have a servers element, just add a new server element with the appropriate id, and password

Configuring Deployment Credentials for Nexus Site Deployment

```
<settings>
  <servers>
    <server>
      <id>nexus-site</id>
      <username>deployment</username>
      <password>deployment123</password>
```

```
</server>
</servers>
</settings>
```

Note

[Configuring Deployment Credentials for Nexus Site Deployment](#), uses the default deployment user and the default deployment user password. You will need to configure the username and password to match the values expected by your Nexus installation.

16.5 Creating a Maven Site Repository

To create a Maven Site Repository, log in as a user with Administrative privileges, and click on "Repositories" under Views.Repositories in the Nexus menu. Under the Repositories tab, click on the Add... drop-down and choose "Hosted Repository" as shown in Figure 16.1.

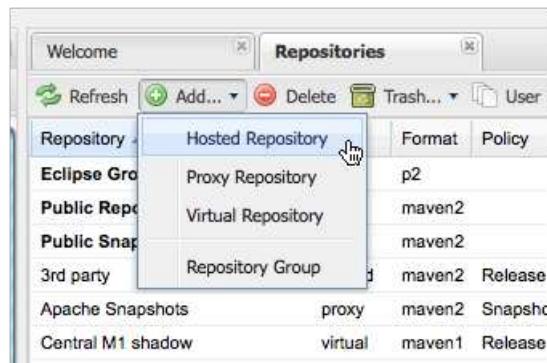


Figure 16.1: Adding a Hosted Repository

In the New Hosted Repository form, click on the Repository Type drop-down and chose the Maven Site Repository format as shown in Figure 16.2. Although you can use any arbitrary name and identifier for your own Nexus repository, for the chapter's example, use a Repository ID of "site" and a Repository Name of "Maven Site".

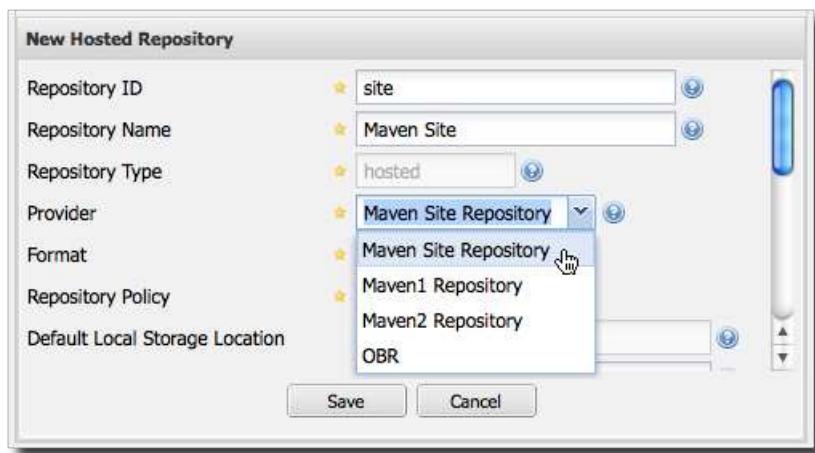


Figure 16.2: Creating a New Maven Site Repository

After creating a new Maven Site repository, it should appear in the list of Nexus repositories as shown in Figure 16.3. Note that the Repository Path shown in Figure 16.3, is the same as the repository path referenced in [Distribution Management for Site Deployment to Nexus](#).

Repository	Type	Format	Repository Status	Repository Path
Maven Central	proxy	maven2	In Service	http://localhost:8081/nexus/content/repositories/central
Maven Site	hosted	maven-site	In Service	http://localhost:8081/nexus/content/sites/site
Releases	hosted	maven2	In Service	http://localhost:8081/nexus/content/repositories/releases

Figure 16.3: Newly Created Maven Site Repository

16.6 Add the Site Deployment Role

In the Maven Settings shown in [Configuring Deployment Credentials for Nexus Site Deployment](#), you configured your Maven instance to use the default deployment user and password. To successfully deploy a site to Nexus, you will need to make sure that the deployment user has the appropriate role and permis-

sions. To add the site deployment role to the deployment user, click on Users under the Security section of the Nexus menu, and then highlight the deployment user as shown in Figure 16.4.

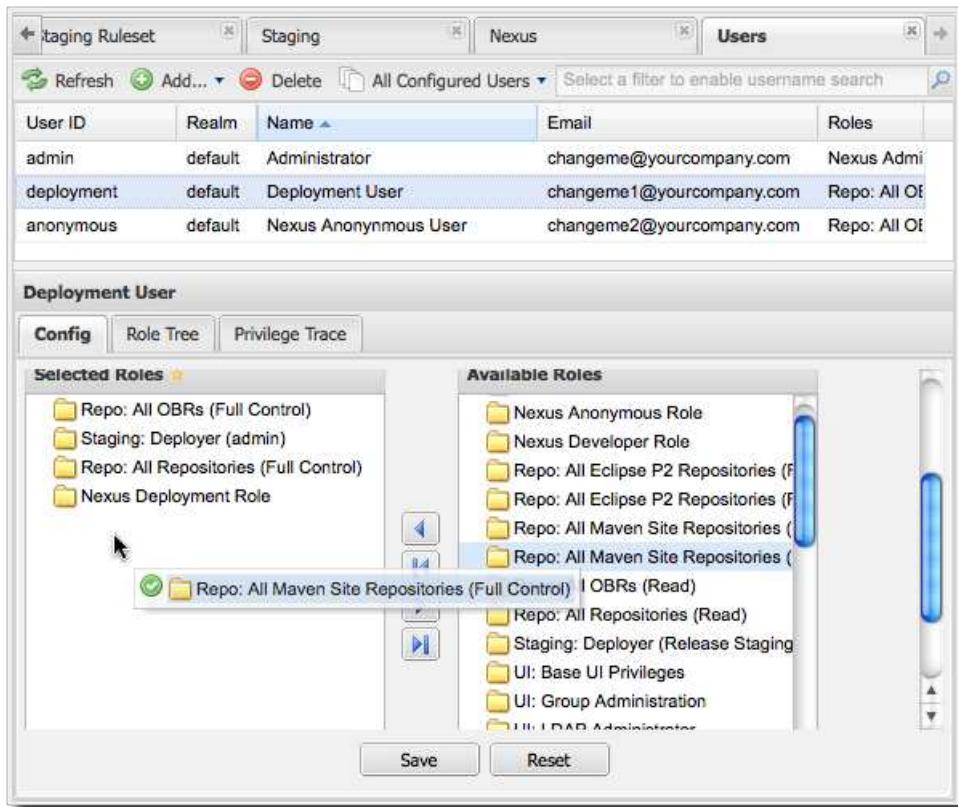


Figure 16.4: Adding the Site Deployment Role to the Deployment User

Locate the "Repo: All Maven Site Repositories (Full Control)" role in the Available Roles list, click this role, and drag it over to the Selected Roles list. Once the role is in the Selected Roles list, click on the Save button to update the roles for the deployment user. The deployment user will now have the ability to publish sites to a Maven Site repository.

16.7 Publishing a Maven Site to Nexus

To publish a site to a Maven Site repository in Nexus Professional, run mvn site site:deploy from the sample-site/ project created earlier in this chapter. The Maven Site plugin will deploy this site to Nexus using the credentials stored in your Maven Settings.

```
~/examples/sample-site$ mvn site site:deploy
[INFO] Scanning for projects...
[INFO]   ↵
-----
[INFO] Building sample-site
[INFO]   task-segment: [site, site:deploy]
[INFO]   ↵
-----
[INFO] [site:site {execution: default-site}]
[INFO] Generating "About" report.
[INFO] Generating "Issue Tracking" report.
[INFO] Generating "Project Team" report.
[INFO] Generating "Dependencies" report.
[INFO] Generating "Project Plugins" report.
[INFO] Generating "Continuous Integration" report.
[INFO] Generating "Source Repository" report.
[INFO] Generating "Project License" report.
[INFO] Generating "Mailing Lists" report.
[INFO] Generating "Plugin Management" report.
[INFO] Generating "Project Summary" report.
[INFO] [site:deploy {execution: default-cli}]
http://localhost:8081/nexus/content/sites/site/ - Session: Opened
Uploading: ./css/maven-base.css to http://localhost:8081/nexus/content/ ↵
sites/site/
#http://localhost:8081/nexus/content/sites/site//./css/maven-base.css \
- Status code: 201

Transfer finished. 2297 bytes copied in 0.052 seconds
Uploading: ./css/maven-theme.css to http://localhost:8081/nexus/content/ ↵
sites/site/
#http://localhost:8081/nexus/content/sites/site//./css/maven-theme.css \
- Status code: 201

Transfer finished. 2801 bytes copied in 0.017 seconds

Transfer finished. 5235 bytes copied in 0.012 seconds
http://localhost:8081/nexus/content/sites/site/ - Session: Disconnecting
```

```
http://localhost:8081/nexus/content/sites/site/ - Session: Disconnected  
[INFO]  ↵  
-----  
[INFO] BUILD SUCCESSFUL  
[INFO]  ↵  
-----  
[INFO] Total time: 45 seconds  
[INFO] Finished at: Sat Oct 03 07:52:35 CDT 2009  
[INFO] Final Memory: 35M/80M  
[INFO] -----
```

Once the site has been published, you can load the site in a browser by going to <http://localhost:8081/nexus/content/sites/site/>

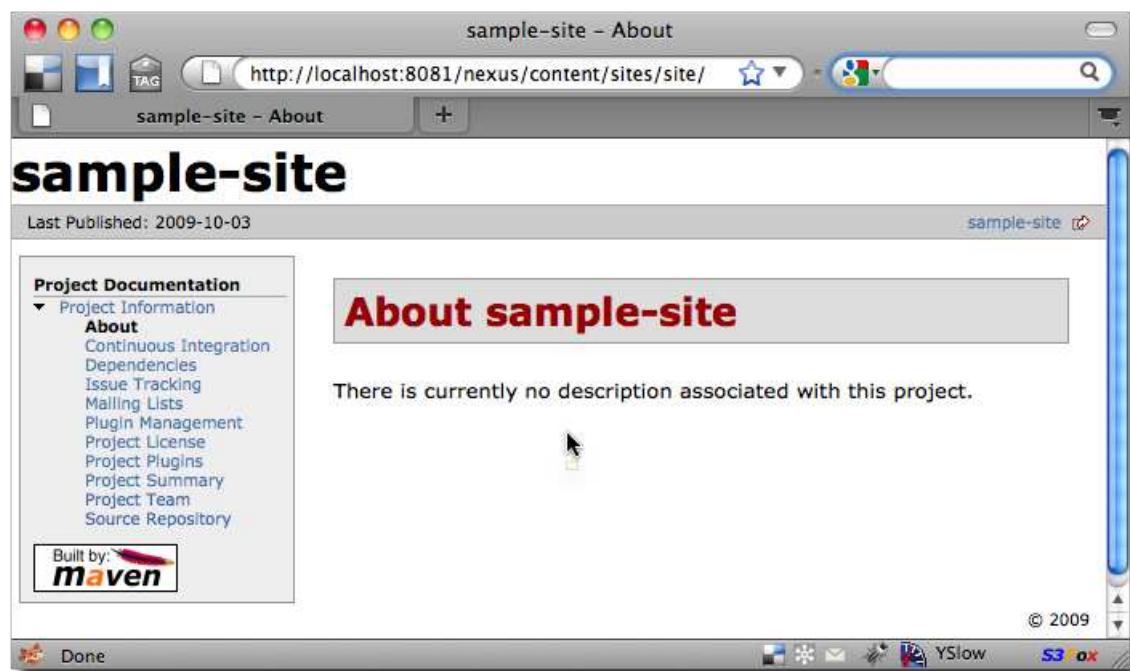


Figure 16.5: Sample Site Maven Project Web Site

Chapter 17

Custom Metadata Plugin

17.1 Introduction

The custom metadata plugin is an optional plugin that comes as part of any Nexus download. It allows you to view, edit and search for additional metadata about any artifact in your Nexus repositories.

The directory containing the plugin code is called `nexus-custom-metadata-plugin-X.Y.Z`. Install the plugin following the instructions in Section [3.12](#).

17.2 Viewing Artifact Metadata

The Custom Metadata Plugin gives you the ability to view artifact metadata. When browsing repository storage or a repository index, clicking on an artifact will load the Artifact Information panel. Selecting the Artifact Metadata tab will display the interface shown in Figure [17.1](#).

The screenshot shows the 'Artifact Metadata' tab of the Nexus interface. At the top, there's a 'Filter: Enter filter term...' input field, an 'Add...' button with a green plus sign, and a 'Delete' button with a red minus sign. Below this is a table with three columns: 'Key', 'Value', and 'Namespace'. The table contains the following data:

Key	Value	Namespace
approved	1	urn:nexus/user#
approvedBy	tobrien	urn:nexus/user#
artifactId	activemq-core	urn:maven#
baseVersion	5.3.0	urn:maven#
extension	jar	urn:maven#
groupId	org.apache.activemq	urn:maven#
path	/org/apache/activemq/activemq-cor...	urn:maven#
repositoryId	central	urn:maven#
type	urn:maven#artifact	http://www.w3.org/1999/02/22-rdf-syntax-ns#
version	5.3.0	urn:maven#

At the bottom of the interface are 'Save' and 'Reset' buttons.

Figure 17.1: Viewing Artifact Metadata

Artifact metadata consists of a key, a value, and a namespace. Existing metadata from an artifact's POM is given a urn:maven namespace, and custom attributes are stored under the urn:nexus/user namespace.

17.3 Editing Artifact Metadata

The Custom Metadata Plugin gives you the ability to add custom attributes to artifact metadata. To add a custom attribute, click on an artifact in Nexus, and select the Artifact Metadata tab. On the Artifact Metadata tab, click on the Add... button and a new row will be inserted into the list of attributes. Supply a key and value and click the Save button to update an artifact's metadata. Figure 17.2 shows the Artifact Metadata panel with two custom attributes: "approvedBy" and "approved".

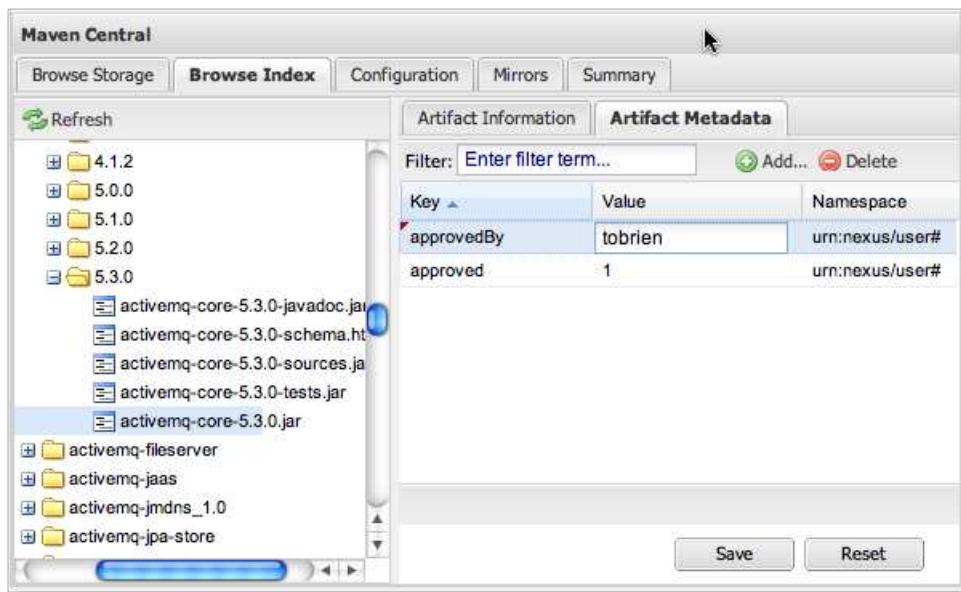


Figure 17.2: Editing Artifact Metadata

17.4 Searching Artifact Metadata

The Custom Metadata Plugin provides you with the ability to configure custom artifact metadata and search for artifacts with specific metadata. To search for artifacts using metadata, click on the Advanced Search link directly below the search field in the Nexus application menu to open the Search panel. Once in the search panel, click on the Keyword Search and click on Metadata Search in the search type drop-down as shown in Figure 17.3.

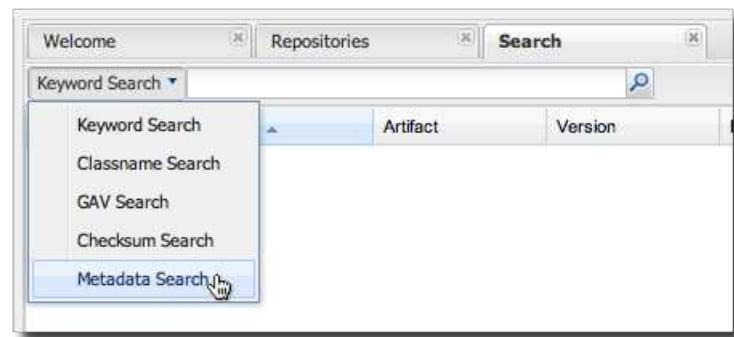


Figure 17.3: Searching Artifact Metadata

Once you select the Metadata Search you will see two search fields and an operator drop-down. The two search fields are the key and value of the metadata you are searching for. The key corresponds to the key of the metadata you are searching for, and the value contains the value or value range you are searching for. The operator drop-down can be set to Equals, Matches, Bounded, or Not Equal.

A screenshot of the Nexus search results for custom metadata. The search interface shows a table of results for artifacts with the key 'approved'. The table columns are 'Source Index', 'Group', 'Artifact', 'Version', 'Packaging', and 'Classifier'. Two records are listed: 'Maven Central (hosted)' with group 'abbot' and artifact 'abbot', version '0.12.3', packaging 'jar'; and 'Maven Central (hosted)' with group 'org.apache.activemq' and artifact 'activemq-core', version '5.3.0', packaging 'jar'. Below the table, it says 'Displaying 2 records' and 'Clear Results'. At the bottom, there are tabs for 'Artifact Information' and 'Artifact Metadata'. Under 'Artifact Information', fields show Group: 'org.apache.act', Artifact: 'activemq-core', Version: '5.3.0', and Download: 'pom, artifact'. Under 'Artifact Metadata', there is an XML field containing the following code:

```
<dependency>
<groupId>org.apache.activemq</groupId>
<artifactId>activemq-core</artifactId>
<version>5.3.0</version>
</dependency>
```

Figure 17.4: Metadata Search Results for Custom Metadata

Once you locate a matching artifact in the Metadata Search interface, click on the artifact and then select the Artifact Metadata to examine an artifacts metadata as shown in Figure 17.5.

The screenshot shows the Nexus Metadata Search interface. At the top, there is a search bar with the key 'approvedBy' and value 'to*'. Below the search bar, a table displays the results: Source Index (Maven Central ...), Group (org.apache.activemq), Artifact (activemq-core), Version (5.3.0), and Packaging (jar). A message below the table says 'Displaying 1 records' and 'Clear Results'. Below this, there are two tabs: 'Artifact Information' (selected) and 'Artifact Metadata'. The 'Artifact Metadata' tab is currently inactive. A filter input field 'Enter filter term...' is present. A table lists the custom metadata for the artifact:

Key	Value	Namespace
approved	1	urn:nexus:user#
approvedBy	tobrien	urn:nexus:user#
artifactId	activemq-core	urn:maven#
baseVersion	5.3.0	urn:maven#
extension	jar	urn:maven#
groupId	org.apache.activemq	urn:maven#
path	/org/apache/activemq/activemq-core/5....	urn:maven#
repositoryId	central	urn:maven#
type	urn:maven#artifact	http://www.w3.org/1999/02/22-rdf-syntax-ns#type
version	5.3.0	urn:maven#

At the bottom of the interface are 'Save' and 'Reset' buttons.

Figure 17.5: Metadata Search Results for Custom Metadata

Chapter 18

User Account Plugin

18.1 Introduction

Nexus Professional's User Account Plugin allows anonymous users to sign-up for a Nexus account without administrative intervention. This "self-serve" capability for user account creation is especially important when an open source project is using Nexus as a primary means for distribution. In such an environment, there may be hundreds of Nexus users who all need a basic level of read-only access, and making these users wait for an administrator to create an account makes little sense. In such a setting, anyone can create an account, activate the account via a verification email, and then access a repository with a default, read-only level of access which you can define.

18.2 Installing the User Account Plugin

The user account plugin is an optional plugin that comes as part of any Nexus download. The directory containing the plugin code is called `nexus-user-account-plugin-X.Y.Z`. Install the plugin following the instructions in Section [3.12](#).

18.3 Configuring the User Account Plugin

To configure the User Account Plugin, click on Server under the Administration section of the Nexus menu, and scroll down to the section named "User Sign Up". The User Sign Up section is shown in Figure 18.1.

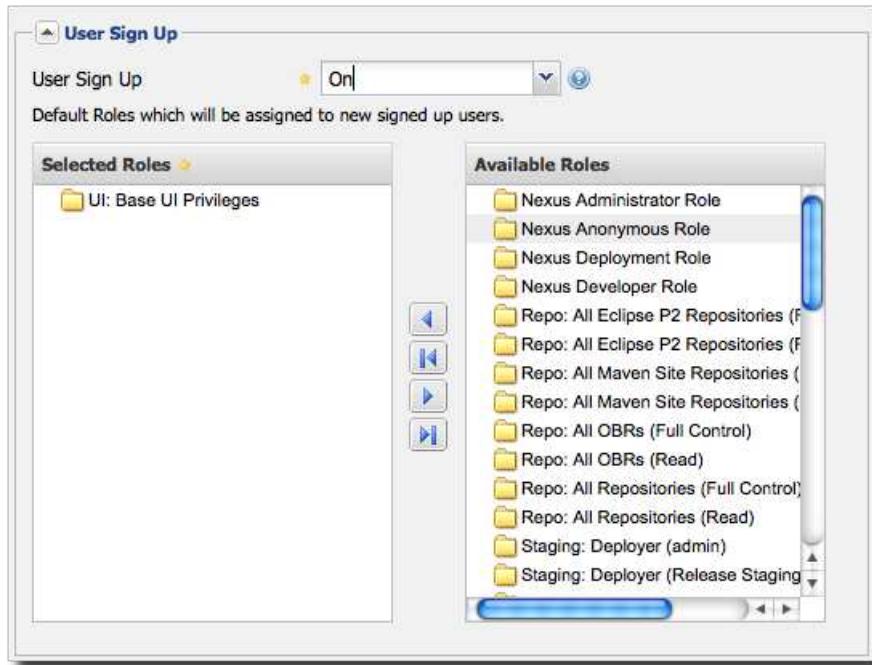


Figure 18.1: Configuring the User Account Plugin

To activate the User Sign Up feature, set the "User Sign Up" feature to "On". This will expose a "Sign Up" link next to the "Log In" link in the Nexus user interface. The Selected Roles in this configuration section are the default roles assigned to users who successfully signed up for an account.

18.4 Signing Up for an Account

Once User Sign Up has been activated via the Server settings as shown in Section 18.3, users will see a Sign Up link next to the Log In link in the Nexus interface. This Sign Up link can be seen in the upper right-hand corner of Figure 18.2.

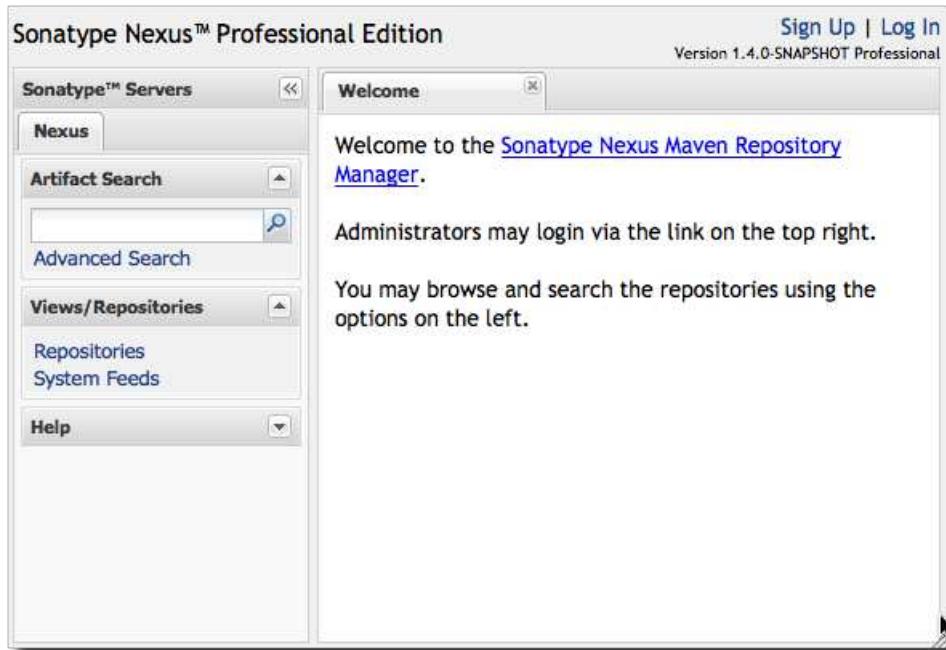


Figure 18.2: Sign Up Link Available for All Nexus Users

Clicking on this Sign Up link will display the Nexus Sign Up dialog shown in Figure 18.3. This form accepts a username, password, the full name of the new user, and an email account. It also asks the users to type in some text from a captcha form element. If a user cannot read the text in the captcha, they can click on the captcha to refresh it with new text.



Figure 18.3: Nexus Sign Up Form

Once the new user clicks on the Sign Up button, they will receive a confirmation dialog which instructs them to check for an activation email.



Figure 18.4: Nexus Sign Up Confirmation

The user will then receive an email containing an activation link. When a user signs up for a Nexus account, the newly created account is disabled until they click on the activation link contained in this email. A sample of the activation email is shown in Figure 18.5.

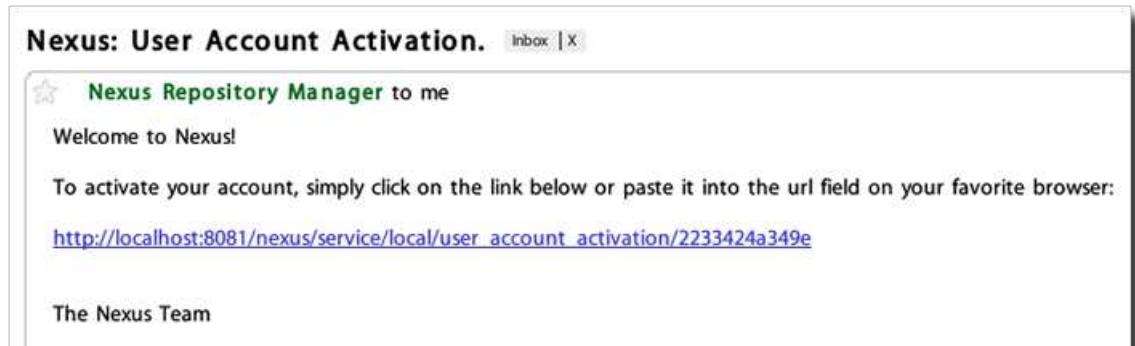


Figure 18.5: Nexus Activation Email

Note

The example activation email in Figure 18.5, points to localhost:8081. You can change this URL by changing the Base URL setting in the Application Server Settings section of the Server configuration. To change this setting, click on the Server link under Administration in the Nexus menu.

18.5 Manual Activation of New Users

If a user does not receive the activation email after signing up for a new account, an Administrator may need to manually activate a new user. To do this, go to the list of Nexus users by clicking on the Users link under Security in the Nexus menu. Locate and select the new user in the list of Nexus users, and change the Status from Disabled to Enabled as shown in Figure 18.6.

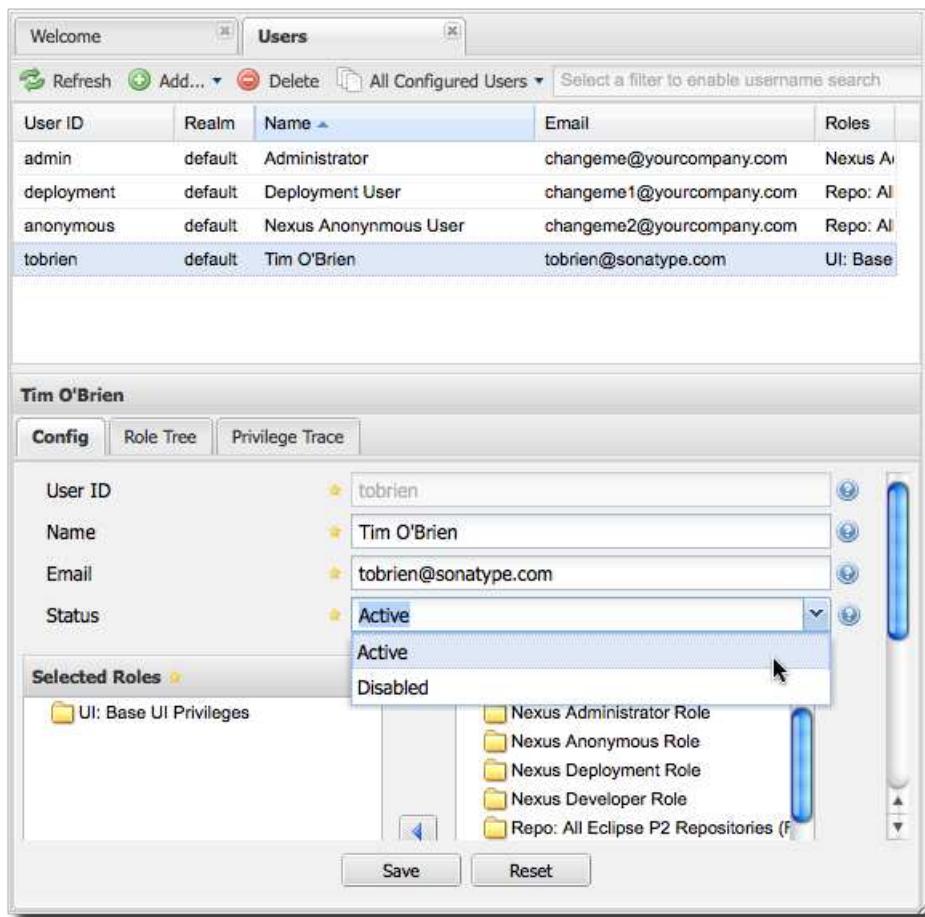


Figure 18.6: Nexus Activation Email

18.6 Modifying Default User Permissions

The default user permissions in the User Sign Up feature only includes "UI: Base UI Privileges". If a user signs up with just this simple permission, the only thing they will be able to do is login, change their password, and logout. Figure 18.7, shows the interface a user would see after logging in with only the base UI privileges.



Figure 18.7: User Interface with only the Base UI Privileges

To provide some sensible default permissions, click on the Server under the Administration section of the Nexus menu and scroll down to the User Sign Up section of the Server settings. Make sure that the selected default roles for new users contain some ability to browse, search, and view repositories.

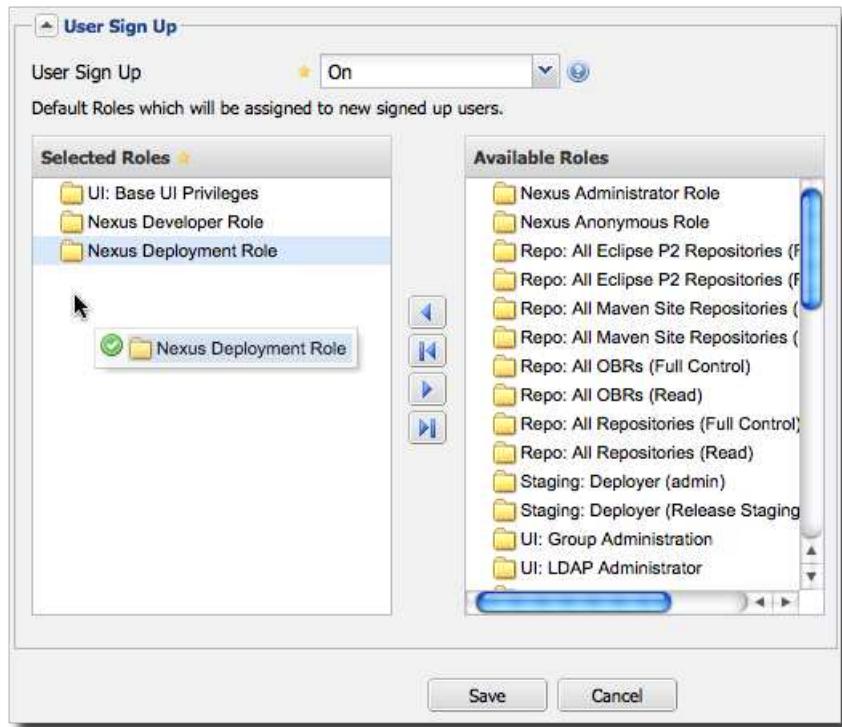


Figure 18.8: Selecting Default Roles for New Users

Warning

Figure 18.8 shows a default User Sign Up role containing the Nexus Deployment Role. If your server were available to the public this wouldn't be a wise default role as it would allow anyone to sign up for an account, activate an account, and start publishing artifacts to hosted repositories with little or no oversight. Such a default role may only make sense if you are running an internal, corporate instance of Nexus Professional and you are comfortable granting any developer in the organization deployment permissions.

Chapter 19

Nexus Atlassian Crowd Plugin

19.1 Introduction

Atlassian's 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. Nexus Professional contains an optional security plugin that allows you to configure Nexus to authenticate against an Atlassian Crowd instance. For more information about Atlassian Crowd, go to <http://www.atlassian.com/software/crowd/>

19.2 Installing the Crowd Plugin

The user account plugin is an optional plugin that comes as part of any Nexus Professional download. The directory containing the plugin code is called enterprise-crowd-plugin-X.Y.Z. Install the plugin following the instructions in Section 3.12.

19.3 Configuring the Crowd Plugin

Once the Atlassian Crowd plugin is installed, restart Nexus and login as a user with Administrative privileges. To configure the Crowd plugin, click on the Crowd Configuration in the Security section of the Nexus menu as shown in Figure 19.1.



Figure 19.1: Crowd Menu Link under the Security Section of the Nexus Menu

Clicking on the Crowd Configuration link will load the form shown in Figure 19.2. This configuration panel contains all of the options that need to be configured to connect your Nexus instance to Crowd for authorization and authentication.

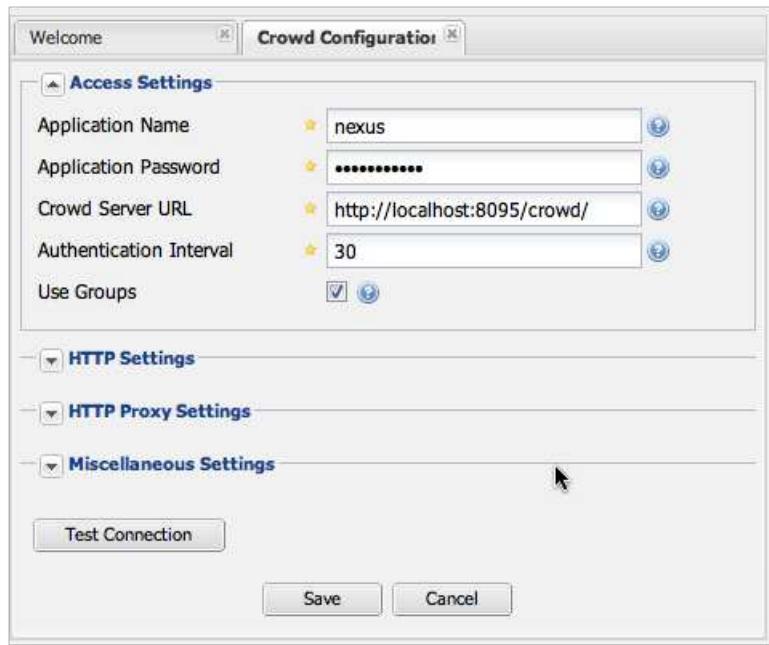


Figure 19.2: Crowd Configuration Panel

The following sections outline all of the settings in the Crowd Configuration Pane.

19.4 Crowd Access Settings

The Access Settings section of the Crowd configuration is shown in Figure 19.3. This section contains the following fields:

Application Name

This field contains the application name of a Crowd application. This value should match the value in the Name field of the form shown in Figure 19.8.

Application Password

This field contains the application password of a Crowd application. This value should match the value in the Password field of the form shown in Figure 19.8.

Crowd Server URL

This is the URL of the Crowd Server, this URL should be accessible to the Nexus process as it is the URL that Nexus will use to connect to Crowd's SOAP services.

Authentication Interval

This is the number of minutes that a Crowd authentication is valid for. This value is in units of minutes, and a value of 30 means that Nexus will only require re-authentication if more than 30 minutes have elapsed since a previously authenticated user has accessed Nexus.

Use Groups

If clicked, Use Groups allows Nexus to use Crowd Groups when calculating Nexus Roles. When selected, you can map a Nexus Role to a Crowd Group.

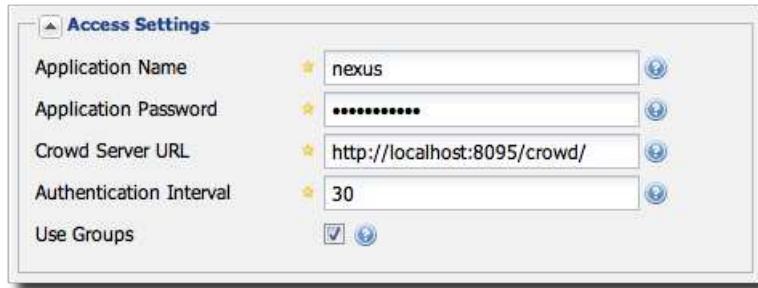


Figure 19.3: Crowd Access Settings

19.4.1 Crowd HTTP Settings

You can control the concurrency of connections to Crowd in the HTTP Settings section shown in Figure 19.4. If you have a high-traffic instance of Nexus, you will want to limit the number of simultaneous connections to the Crowd server to a reasonable value like 20. The HTTP Timeout specifies the number of milliseconds Nexus will wait for a response from Crowd. A value of zero for either of these properties indicates that there is no limit to either the number of connections or the timeout.

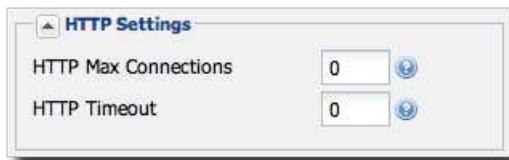


Figure 19.4: Crowd HTTP Settings

19.4.2 Crowd HTTP Proxy Settings

If your Nexus installation is connecting to Crowd via an HTTP Proxy server, the HTTP Proxy Settings section of the Crowd Configuration allows you to specify the host, port, and credentials for a HTTP Proxy server. The HTTP Proxy Settings section is shown in Figure 19.5.

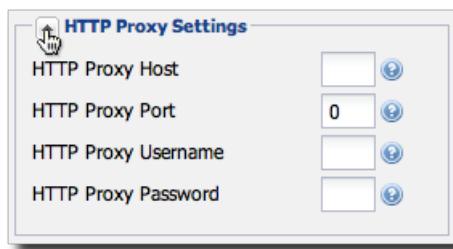


Figure 19.5: Crowd HTTP Proxy Settings

19.4.3 Miscellaneous Settings

The miscellaneous settings section shown in Figure 19.6, allows you to configure settings that control the name of the Single Sign-on cookie and the various keys that are used to retrieve values that relate to authentication and the auth token. This dialog is only relevant if you have modified optional Crowd settings in your \$CROWD_HOME/etc/crowd.properties. For more information about customizing these options see the [Atlassian Crowd documentation](#)

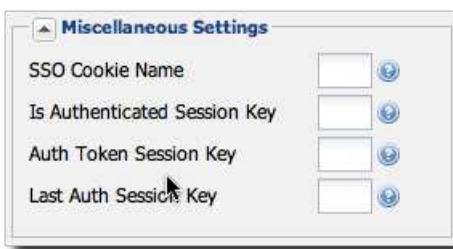


Figure 19.6: Crowd Miscellaneous Settings

19.5 Adding the Crowd Authentication Realm

Once you have configured Nexus to connect to Crowd, you must select the Crowd authorization realm from the list of available realms in your Nexus Server settings. Figure 19.7, shows the Security settings section in the Nexus Server configuration panel. To load the Nexus server configuration panel, click on Server under Administration in the Nexus menu. Drag Crowd from the list of available realms to the list of selected realms and then save the Nexus server configuration.

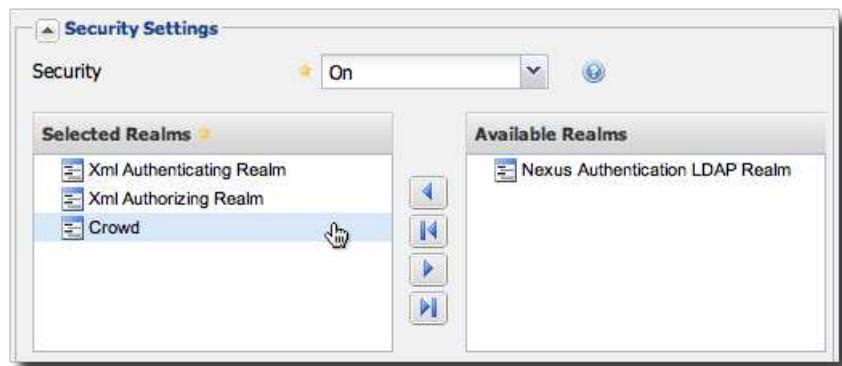


Figure 19.7: Configuring the Crowd Authentication Realm

19.6 Configuring a Nexus Application in Crowd

To connect Nexus to Atlassian's Crowd, you will need to configure Nexus as an application in Crowd. To do this, login to Crowd as a user with Administrative rights, and click on the Applications tab. Once you click on this tab, you should see two options under the Applications tab: Search Applications and Add Application. Click on Add Application to display the form shown in Figure 19.8, and create a new application with the following values in the Details tab of the Add Application form:

- Application Type: Generic Application
- Name: nexus
- Description: Sonatype Nexus Professional

Choose a password for this application. Nexus will use this password to authenticate with the Crowd server. Click on the Next button.

The screenshot shows a 'Add Application' dialog box with five tabs at the top: 1. Details (selected), 2. Connection, 3. Directories, 4. Authorisation, and 5. Confirmation. The 'Application Type:' dropdown is set to 'Generic Application'. Below it is a note: 'Are you connecting JIRA to Crowd, or perhaps Confluence or Bamboo?' The 'Name:' field contains 'nexus'. The 'Description:' field contains 'Sonatype Nexus Professional' with a note: 'A short description of the application. Often a URL is helpful.' The 'Password:' and 'Confirm Password:' fields both contain masked text. At the bottom are 'Next >' and 'Cancel' buttons.

Figure 19.8: Creating a Nexus Crowd Application

Clicking on Next will advance the form to the Connection tab shown in Figure 19.9. In this tab you need to supply the URL Nexus and the remote IP address for Nexus. Figure 19.9, shows the Connection form configured for a local instance of Nexus. If you were configuring Crowd and Nexus in a production environment, you would supply the URL that users would use to load Nexus in a web browser and you would supply the IP address that Nexus will be connecting from. Once you have completed the Connection form, click on Next to advance to the Directories form.

The screenshot shows the 'Add Application - nexus' dialog with the 'Connection' tab selected. It has five tabs in total: '1. Details', '2. Connection', '3. Directories', '4. Authorisation', and '5. Confirmation'. The 'Connection' tab is active, indicated by a blue border. The 'URL:' field contains 'http://localhost:8081/nexus' with a red asterisk indicating it is required. A 'Resolve IP Address' button is available. Below the URL field is a note: 'The URL where this application resides, e.g. http://jira.atlassian.com'. The 'Remote IP Address:' field contains '127.0.0.1' with a red asterisk, and a 'Cancel' button is shown to its right. Below the IP address field is a note: 'The IP address for the application, e.g. 127.0.0.1'. At the bottom are 'Next >' and 'Cancel' buttons.

Figure 19.9: Creating a Nexus Crowd Application Connection

Clicking on Next advances to the Directories form shown in Figure 19.10. In this example, the Nexus application in Crowd is going to use the default "User Management" directory. Click on the Next button to advance to the "Authorisation" form.

The screenshot shows the 'Add Application - nexus' dialog with the 'Directories' tab selected. It has five tabs in total: '1. Details', '2. Connection', '3. Directories', '4. Authorisation', and '5. Confirmation'. The 'Directories' tab is active, indicated by a blue border. A note at the top says: 'select the directories you are going to let this application use for authentication and authorisation.' Below this, under 'User Management:', there is a checked checkbox labeled 'Crowd Internal Directory – This is a user management directory'. At the bottom are 'Next >' and 'Cancel' buttons.

Figure 19.10: Creating a Nexus Crowd Application Directories

Clicking on the Next button advances to the "Authorisation" form shown in Figure 19.11. If any of the directories selected in the previous form contain groups, each group is displayed on this form next to a checkbox. You can select "Allow all users" for a directory, or you can select specific groups which are allowed to authenticate to Crowd through Nexus. This option would be used if you wanted to limit Nexus access to specific subgroups within a larger Crowd directory. If your entire organization is stored in a

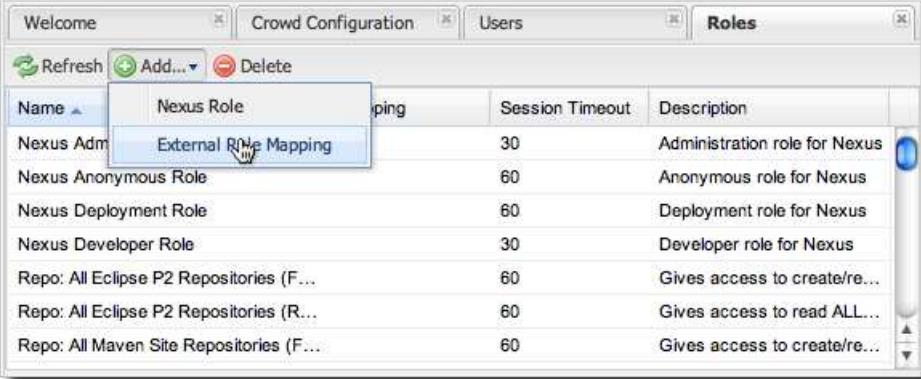
single Crowd directory, you may want to limit Nexus access to a group that contains only Developers and Administrators.



Figure 19.11: Creating a Nexus Crowd Application Authorization

19.7 Mapping Crowd Groups to Nexus Roles

To map a Crowd Group to a Nexus Role, open up the Roles panel by clicking on the Roles link under the Security section of the Nexus menu. Click on the Add... button and select External Role Mapping as shown in Figure 19.12.



The screenshot shows a table titled 'Roles' with columns: Name, Nexus Role, Mapping, Session Timeout, and Description. The 'Name' column is sorted by clicking the arrow icon. A row for 'Nexus Admin' has its 'Mapping' field highlighted with a blue selection bar, indicating it is selected for modification.

Name	Nexus Role	Mapping	Session Timeout	Description
Nexus Admin	External Role Mapping		30	Administration role for Nexus
Nexus Anonymous Role			60	Anonymous role for Nexus
Nexus Deployment Role			60	Deployment role for Nexus
Nexus Developer Role			30	Developer role for Nexus
Repo: All Eclipse P2 Repositories (F...			60	Gives access to create/re...
Repo: All Eclipse P2 Repositories (R...			60	Gives access to read ALL...
Repo: All Maven Site Repositories (F...			60	Gives access to create/re...

Figure 19.12: Adding an External Role Mapping

Selecting External Role Mapping will show the Map External Role dialog shown in Figure 19.13.

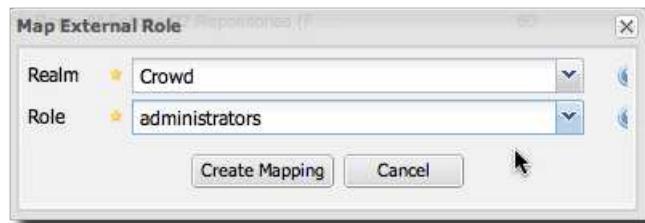


Figure 19.13: Mapping an External Crowd Group to a Nexus Role

Once you have mapped a Crowd Group to a Nexus Role, these Roles will appear in the list of Nexus Roles with a mapping value of "Crowd" as shown in Figure 19.14.

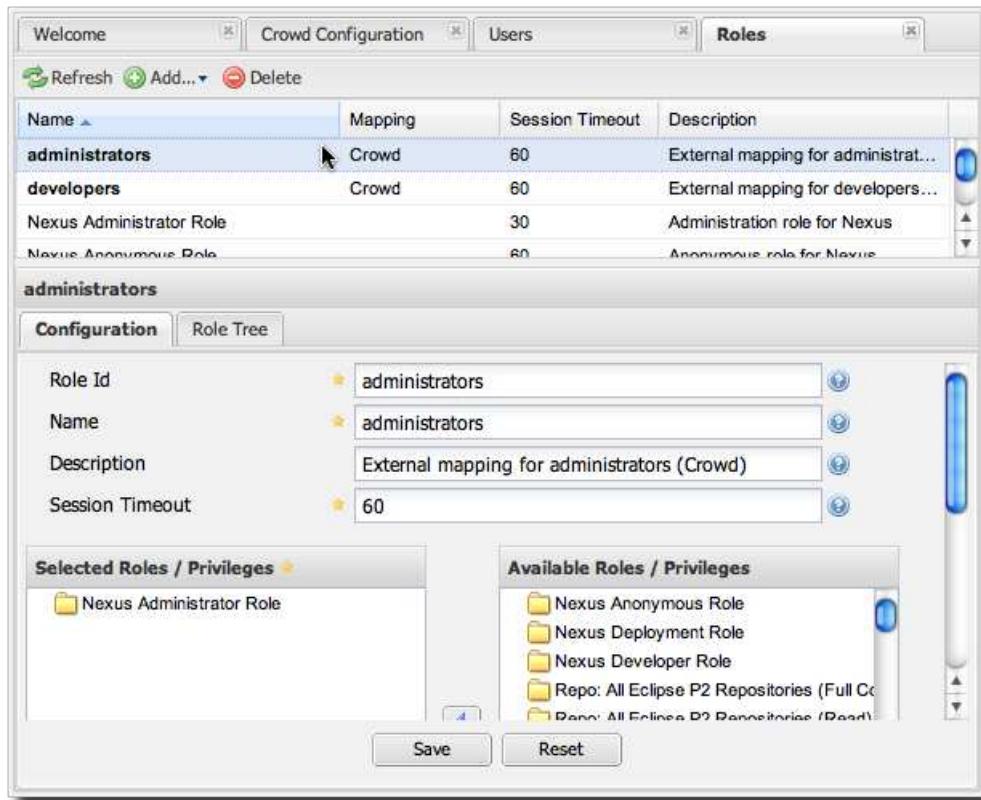


Figure 19.14: Two Crowd Groups Mapped to Nexus Roles

19.8 Adding a Crowd Role to a Nexus User

To illustrate this feature, consider the crowd-manager user with an id of "brian". This user's groups are shown in Figure 19.15.

The screenshot shows the Crowd web interface. At the top, there's a navigation bar with links for Applications, Users, Groups, Roles, Directories, and Administration. The User: Default Admin link is visible. Below the navigation bar, on the left, is a sidebar with links for Search Users, Add User, Import Users, Reset Password, and Remove User. The main content area is titled "View User – brian". It has tabs for Details, Attributes, Groups (which is selected), Roles, and Applications. A sub-section titled "These are the groups the user is a member of." displays a table with one row: Group (developers), Description (Developers), and Active (true). At the bottom right of this section are "Add Groups" and "Remove Groups" buttons.

Figure 19.15: Crowd Groups for User "brian"

To add an external user role mapping, open up the Users panel by clicking on Users in the Security section of the Nexus panel. Click on the Add... button and select External User Role Mapping from the drop-down as shown in Figure 19.16.

The screenshot shows the Nexus interface with a toolbar at the top featuring Welcome, Crowd Configuration, Users, and Roles. Below the toolbar is a search bar and a menu for selecting a filter. The main content area is a table with columns for User ID, Name, Email, and Roles. There are three rows: admin (Nexus User, Email: changeme@yourcompany.com, Roles: Nexus Admin...), deployment (Deployment User, Email: changeme1@yourcompany.com, Roles: Repo: All O...), and anonymous (Nexus Anonymous User, Email: changeme2@yourcompany.com, Roles: Repo: All O...). The "admin" row is selected. A dropdown menu is open over the "Name" column of the "admin" row, with "External User Role Mapping" highlighted and a cursor pointing at it.

Figure 19.16: Adding an External User Role Mapping

Selecting External User Role Mapping will show the dialog shown in Figure 19.17.



Figure 19.17: Locating a Crowd User in the User Role Mapping Dialog

Once you locate the Crowd user that you want to add a Nexus Role to... You can use the configuration panel shown in Figure 19.18, to add a Role to the Crowd-managed "brian" user.

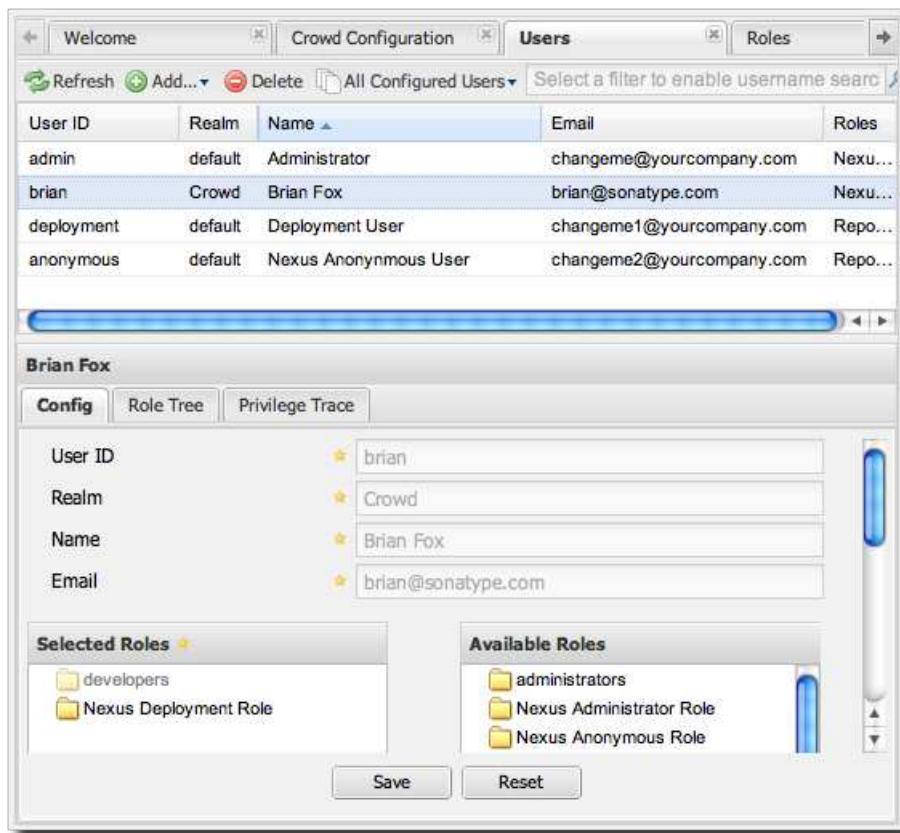


Figure 19.18: Adding a Nexus Role to a Crowd User

Chapter 20

Artifact Bundles

20.1 Introduction

Artifact bundles are groups of related artifacts which are all related by the same groupId, artifactId, and version (GAV) coordinate. They are used by projects that wish to upload artifacts to the Central Repository.

Bundles must contain the following POM elements:

- modelVersion
 - groupId
 - artifactId
 - packaging
 - name
 - version
 - description
 - url
 - licenses
-

- scm
 - url
 - connection

20.2 Creating an Artifact Bundle from a Maven Project

Artifact bundles are created with the Maven Repository Plugin. For more information about the Maven Repository plugin, see <http://maven.apache.org/plugins/maven-repository-plugin/>

[Sample POM Containing all Required Bundle Elements](#), lists a project's POM which satisfies all of the constraints that are checked by the Maven Repository plugin. The following POM contains, a description and a URL, SCM information, and a reference to a license. All of this information is required before an artifact bundle can be published to the Maven Central repository.

Sample POM Containing all Required Bundle Elements

```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.sonatype.sample</groupId>
  <artifactId>sample-project</artifactId>
  <packaging>jar</packaging>
  <version>1.0</version>
  <name>sample-project</name>
  <description>A Sample Project for the Nexus Book</description>
  <url>http://books.sonatype.com</url>
  <licenses>
    <license>
      <name>The Apache Software License, Version 2.0</name>
      <url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>
      <distribution>repo</distribution>
    </license>
  </licenses>
  <scm>
    <connection>
      scm:git:git://github.com/sonatype/sample-project.git
    </connection>
    <url>scm:git:git://github.com/sonatype/sample-project.git</url>
    <developerConnection>
      scm:git:git://github.com/sonatype-sample-project.git
    </developerConnection>
  </scm>

```

```
</developerConnection>
</scm>
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
</project>
```

To create a bundle from a Maven project, run the repository:bundle-create goal. This goal will check the POM to see if it complies with the standards for publishing a bundle to a public repository, it will then bundle all of the artifacts generated by a particular build. To build a bundle that only contains the standard, unclassified artifact from a project, run mvn repository:bundle-create. To generate a bundle which contains more than one artifact, run mvn javadoc:jar source:jar repository:bundle-create

```
~/examples/sample-project$ mvn javadoc:jar source:jar repository:bundle- ↵
  create
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'javadoc'.
[INFO]   ←
----- ←
[INFO] Building sample-project
[INFO]   task-segment: [javadoc:jar, source:jar, repository:bundle-create ←
  ]
[INFO]   ←
----- ←
[INFO] [javadoc:jar {execution: default-cli}]
Loading source files for package com.sonatype.sample...
Constructing Javadoc information...
Standard Doclet version 1.6.0_15
Building tree for all the packages and classes...
...
[INFO] Preparing source:jar
[INFO] No goals needed for project - skipping
[INFO] [source:jar {execution: default-cli}]
...
...
```

TESTS

```
Running com.sonatype.sample.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.03 sec
```

```
Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] [jar:jar {execution: default-jar}]
[INFO] Building jar: ~/temp/sample-project/target/sample-project-1.0.jar
[INFO] [repository:bundle-create {execution: default-cli}]
[INFO] The following files are marked for inclusion in the repository ←
  bundle:

0.) Done
1.) sample-project-1.0.jar
2.) sample-project-1.0-javadoc.jar
3.) sample-project-1.0-sources.jar

Please select the number(s) for any files you wish to exclude, or '0' when ←
  \
you're done. Separate the numbers for multiple files with a comma (',').

Selection:
0
[INFO] Building jar: ~/temp/sample-project/target/sample-project-1.0- ←
  bundle.jar
[INFO] ←
----- ←

[INFO] BUILD SUCCESSFUL
[INFO] ←
----- ←

[INFO] Total time: 11 seconds
[INFO] Finished at: Sat Oct 10 21:24:23 CDT 2009
[INFO] Final Memory: 36M/110M
[INFO] ←
----- ←
```

Once the bundle has been created, there will be a bundle JAR in the target/ directory. As shown in the following command output, the bundle JAR contains: a POM, the project's unclassified artifact, the javadoc artifact, and the sources artifact.

```
~/examples/sample-project$ cd target
~/examples/sample-project/target$ jar tvf sample-project-1.0-bundle.jar
0 Sat Oct 10 21:24:24 CDT 2009 META-INF/
98 Sat Oct 10 21:24:22 CDT 2009 META-INF/MANIFEST.MF
1206 Sat Oct 10 21:23:46 CDT 2009 pom.xml
2544 Sat Oct 10 21:24:22 CDT 2009 sample-project-1.0.jar
20779 Sat Oct 10 21:24:18 CDT 2009 sample-project-1.0-javadoc.jar
```

```
891 Sat Oct 10 21:24:18 CDT 2009 sample-project-1.0-sources.jar
```

20.3 Uploading an Artifact Bundle to Nexus

To upload an artifact bundle to Nexus Professional, select Staging Upload from the Enterprise section of the Nexus menu as shown in Figure 20.1.



Figure 20.1: Staging Upload Link

Selecting the Staging Upload link will load the Staging Upload dialog. Choose Artifact Bundle form the Upload Mode drop-down the Staging Upload panel will switch to the form shown in Figure 20.2. Click on Select Bundle to Upload... and then select the JAR that was created with the Maven Repository plugin used in the previous sections. Once a bundle is selected, click on Upload Bundle.

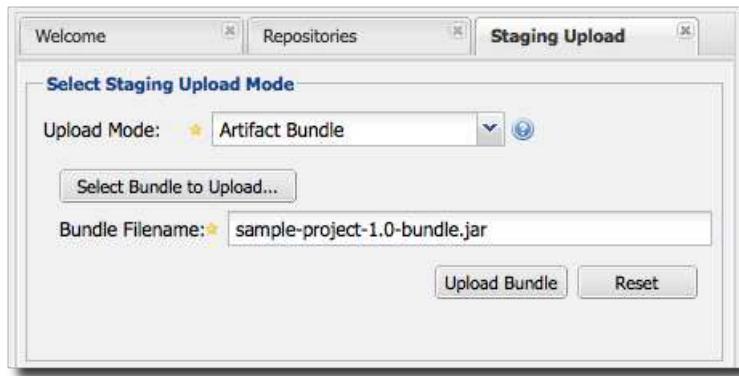


Figure 20.2: Uploading an Artifact Bundle

After uploading the Artifact Bundle, click on the Staging link in the Enterprise section of the Nexus menu as shown in Figure 20.1. Click on Staging will load a list of all the defined staging profiles. Click on the first default profiles named "Release Staging Profile", and you should see that the Staging Artifact Upload created and closed a new staging repository as shown in Figure 20.3. This repository contains all of the artifacts contained in the uploaded bundle. It allows you to promote or drop the artifacts contained in a bundle as a single unit.

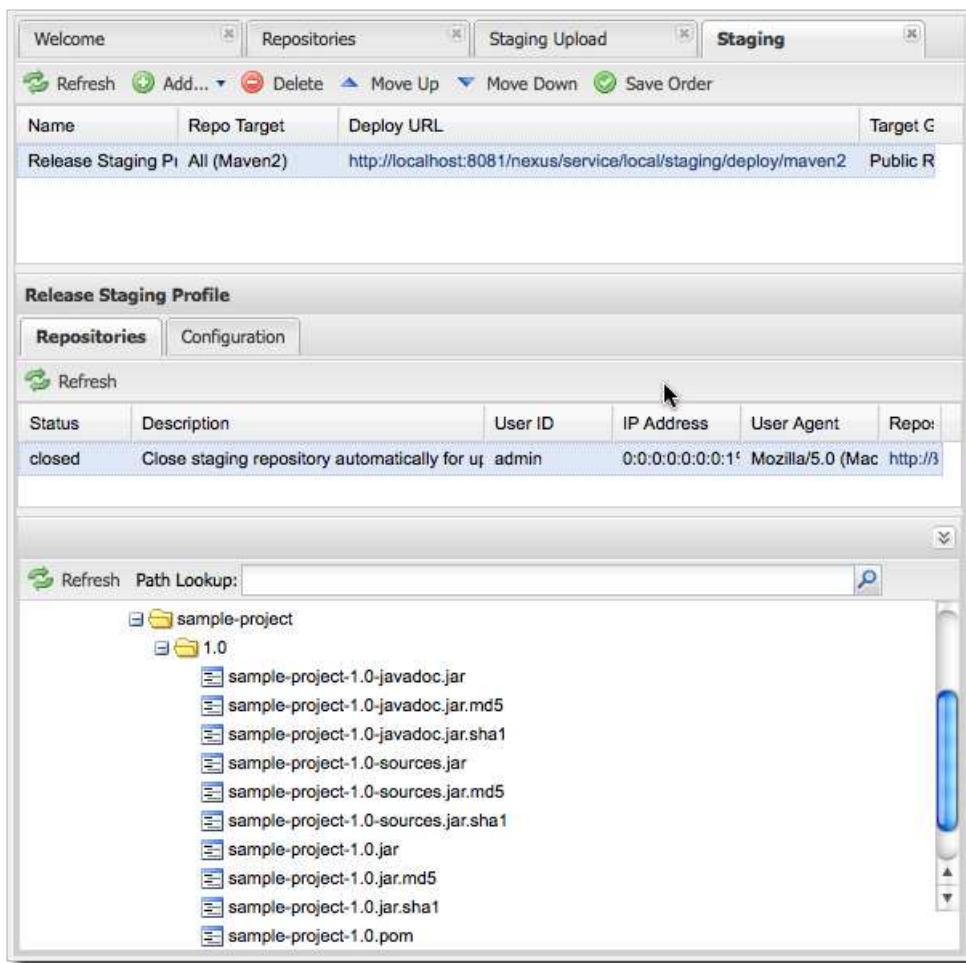


Figure 20.3: Staging Repository Created for Artifact Bundle Upload

To promote a staging repository, right click on the closed repository shown in Figure 20.3, and select "Promote". Once you select promote, Nexus will present the Promote Staged Repository dialog shown in Figure 20.4. Select the repository that you want to published the staging repository to, and click on the Promote button.

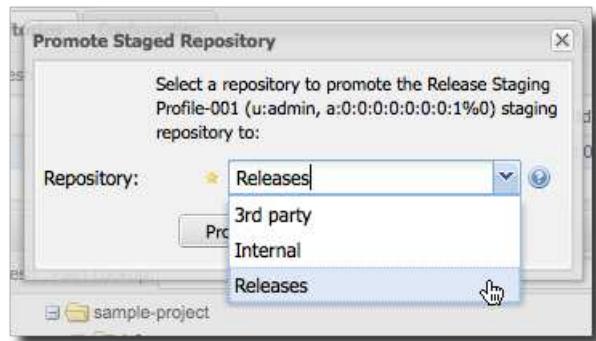


Figure 20.4: Promoting a Staged Repository

After promotion, Nexus will then display a confirmation that the promotion was completed successfully as shown in Figure 20.5.



Figure 20.5: Confirmation that Staged Repository has been Promoted

Chapter 21

Nexus Best Practises

21.1 Introduction

Once you decide to install a Repository Manager, the next decision is how to setup your repositories, particularly if you have multiple teams sharing the same instance. Nexus is very flexible in this area and supports a variety of configurations. I'll first describe the options and then discuss the thought process used to decide what makes sense for your organization.

21.2 Repositories per Project/Team

The first and most obvious way to support multiple teams is to configure a pair of repositories per team (one release, one snapshot). The team is then given the appropriate C.R.U.D. permissions and they are able to use the system for their artifacts.

Our <http://oss.sonatype.org> instance is for the most part configured in this manner, where each project like Jetty has their own repositories separate from everyone else.

21.3 Partition Shared Repositories

Another option is to have a single (or a few) pair of Release/Snapshot repositories for your entire organization. In this case, the access is controlled by a mechanism we call "Repository Targets."

Simply put, a Repository Target is a way to manage a set of artifacts based on their paths. A Repository Target is simply a list of regular expressions and a Name. For example, a Repo Target for Maven would be `./org/apache/maven/`. and Nexus OSS would be `./org/sonatype/nexus/`.

Note

While it is most common to manage artifacts based on the path of their groupId, the Regular Expression is matched against the entire path, and so it is also possible, for example, to define "Sources" as `.*-sources.jar` . . . it's also worth noting that Repository Targets are not mutually exclusive. It is perfectly valid for a given path to be contained by multiple targets.

In this model, you would create a Repo Target for each project in your system. You are then able to take the Repo Target and associate it with one or more Repositories or Groups in your system. When you do this, new, specific, C.R.U.D. privileges are created. For example, I could take the Maven Repo target, associate it with my Release and Snapshot repository, and now I get privileges I can assign to Create, Read, Update, Delete "Maven" (`./org/apache/maven/`) artifacts in my Release and Snapshot repositories.

This method is used to manage the <http://repository.apache.org> instance, where we have just one Release and Snapshot repository and each project team gets permissions to their artifacts based on the path.

21.3.1 Selecting an Approach

First of all, these choices aren't mutually exclusive. In fact, the first option builds upon the default Repository Target of `".*"` which simply gives you access to all artifacts regardless of the path. You still associate the default Repo Target with specific repositories to create the assignable privileges

In general, it's my opinion that fewer repositories will scale better and are easier to manage. It's also easier to start off with a single pair of repositories, with the default "All M2″ (`.*`) target and simply refine the permissions as you scale. Most things that are configured per repository (Cache, Storage location, Snapshot purging, etc) will generally be applicable for all projects, so this mode avoids the duplication of these tasks. Since everything will be stored together in a single folder on disk, it makes backups easier as well.

The reasons why you would want multiple sets of repositories is essentially the opposite of above: If you need different expiration, Snapshot purging or storage folders, then a single shared repo won't work. Replication and fail-over strategies may also make this method easier to support. If you absolutely must maintain total separation between Project teams, i.e. they can't read each other's artifacts, then this solution might be more applicable as well. (but is still possible with Repo Targets...just grant Read to only the appropriate targets)

In Summary, Nexus allows you to control the security of your artifacts based on the repository and/or the path of the artifact, meaning it is possible to slice and dice the system any way you see fit. My default position is to use a few Hosted Repositories as possible and control the permissions by the Repository Target.

Chapter 22

Developing Nexus Plugins

Among the many benefits of using an technology with an open source core is the ability to customize behaviour and create extensions. To this end, Sonatype has spent a great deal of time designing an intuitive Plugin API that will allow you to take Nexus where you need it to go. This chapter summarizes some of these extension points and presents a walk through of how you would start to develop your own Nexus plugins.

Our community has already created a number of compelling and useful plugins, some of which have been integrated into the set of plugins that are distributed with both Nexus Open Source and Nexus Professional. Sonatype tried to make the Plugin API as lightweight and extensible as possible with the following goals in mind:

- Providing a clear set of extension points for plugin developers
- Providing isolated plugin classpaths to avoid compatibility issues between plugins and to prevent a plugin from disturbing another, unrelated part of Nexus.
- Giving developers the ability to load and unload Nexus plugins at runtime

22.1 Nexus Plugins

The Nexus API is contained in a module named `nexus-api`. If you are developing a Nexus plugin, you will need to familiarize yourself with the extension points that are defined in this project.

22.1.1 Nexus Plugin API

Nexus provides an extra module for plugin developers - the "nexus-plugin-api". This module provides some extra annotations for plugins developers, and it allows a plugin developer to implement a plugin without having to know anything about Plexus or Nexus internals.

The Nexus Plugin API uses the `@javax.inject.Inject` annotation, an emerging standard for dependency injection which allows Nexus plugins to be developed in a way that is container-neutral.

The plugin API also introduces some additional annotations to make things easier:

```
@org.sonatype.plugin.Managed
```

When a `@Managed` annotation is present on an interface, it marks the interface as "component contract" (Plexus role). Any non-abstract classes implementing it will be made managed by current container.

```
@org.sonatype.nexus.plugins.RepositoryType
```

Used on interfaces, to mark it as new repository type, and to be registered with other core repository types in Nexus Repository Type Registry. It holds the basic information about the new type (the path where to mount it).

```
@org.sonatype.nexus.plugins.RestResource
```

Used on classes, to mark them as REST Resources.

22.2 Nexus Extension Points

The simplest Nexus plugin contain a single class, SampleEventInspector, which contributes an EventInspector to the Nexus Application. This simple event inspector will do nothing more than print a message every time it accepts and inspects an event.

A Simple Event Inspector

```
package org.sample.plugin;

import org.sonatype.nexus.proxy.events.EventInspector;
import org.sonatype.plexus.appevents.Event;

public class SampleEventInspector implements EventInspector {
    public boolean accepts( Event<?> evt ) {
        return true;
    }

    public void inspect( Event<?> evt ) {
        System.out.println( "Invoked with event: " +
            evt.toString() + " with sender " +
            evt.getEventSender().toString() );
    }
}
```

During the build of this nexus plugin, this class is compiled and then scanned for concrete classes that implement extension point interfaces defined in the following section. The EventInspector interface in the nexus-api project has been marked with the @ExtensionPoint annotation. The plugin build takes the @ExtensionPoint, @Named, and @Inject annotations that may be present and generates a plugin descriptor which is packaged in the plugin's JAR.

When the plugin is present in Nexus during start-up, the Nexus plugin manager reads the plugin metadata and instantiates the appropriate components. To implement a plugin, you simply implement some of these interfaces.

22.3 Nexus Plugin Extension Points

The following sections outline the available Nexus extension points.

22.4 Nexus Plugin Extension

Interface: org.sonatype.nexus.plugins.NexusPlugin

This extension component is meant to be used in Nexus plugins only. If it is found in a plugin, it will be invoked during install/uninstall/init phases of a plugin installation/uninstallation/initialization. Typical usage would be a need to perform some specific tasks on plugin install (i.e. it uses native code to do some magic and those needs to be copied somewhere, register them with OS, etc).

22.5 Nexus Index HTML Customizer

Interface: org.sonatype.nexus.plugins.rest.NexusIndexHtmlCustomizer

This extension is able to customize the "index.html" returned by Nexus. Using this component, a plugin is able to add markup or Javascript to the pages generated by the Nexus web application. Every plugin that has a UI component uses this extension point to add Javascript customizations to the interface.

22.6 Static Plugin Resources

Interface: org.sonatype.nexus.plugins.rest.NexusResourceBundle

This extension gathers and publishes static resources over HTTP. These resources are usually JavaScript files, CSS files, images, etc. Plugin developers do not need to use this extension directly since some of the features it exposes are automatic for all plugins. When the Nexus plugin manager discovers resources in plugin JAR under the path "/static", the Plugin Manager will create a special "plugin NexusResourceBundle" component on the fly.

If you do not want the plugin manager to automatically add a resource bundle you can define your own resource bundle implementation. The plugin manager will not add a resource bundle if:

- no resources found on "/static" path within plugin classpath, or
 - a user created component of NexusResourceBundle exists within plugin
-

The "default plugin" resource bundle component uses MimeUtil from core to select MIME types of resources found within plugin, and will use same path to publish them (i.e. in plugin JAR "/static/image.png" will be published on "http://nexushost/nexus/static/image.png").

22.7 Plugin Templates

Interface: org.sonatype.nexus.templates.TemplateProvider

Template provider is a component providing repository templates to Nexus. Every plugin which provides a "new" repository type should add a TemplateProvider as it is the only way to instantiate a repository instance. The core of Nexus provides a "default" template provider with templates for all core repository types, and all custom repository plugins (P2, OBR) provide template providers for their types.

22.8 Event Inspectors

Interface: org.sonatype.nexus.proxy.events.EventInspector

Event inspectors are used to inspect events in Nexus. One example of where this extension point is used is the index generation. To generate a Nexus index, there is an event inspector which listens for RepositoryItemEvent subclasses and updates the index in response to repository activity.

22.9 Content Generators

Interface: org.sonatype.nexus.proxy.item.ContentGenerator

A content generator is a component that is able to generate content dynamically, on the fly, instead of just serving a static resource. The content generator is registered to respond to a path that corresponds to a file. When the resource is retrieved, Nexus discards the file content and uses the registered content generator to generate content. The Nexus Archetype plugin uses a content generator to generate the archetype-catalog.xml. Every time a client requests the archetype-catalog.xml, the archetype catalog is generated using information from the index.

22.10 Content Classes

Interface: org.sonatype.nexus.proxy.registry.ContentClass

Content class controls the compatibility between repository types. It defines the type of content that can be stored in a repository, and it also affects how repositories can be grouped into repository groups. Every plugin contributing a new repository type should provide an instance of this extension point. Nexus has a ContentClass implementation for every core supported repository type, and the P2 and OBR plugins define custom ContentClass implementations.

22.11 Storage Implementations

Interface: org.sonatype.nexus.proxy.storage.local.LocalRepositoryStorage

Interface: org.sonatype.nexus.proxy.storage.remote.RemoteRepositoryStorage

A plugin developer can override the default file-based local repository storage and the default remote HTTP repository storage interface. If your plugin needs to stores repository artifacts and information in something other than a filesystem, or if your remote repository isn't accessible via HTTP, your plugin would provide an implementation of one of these interfaces. Nexus provides one of the each: a file-system LocalRepositoryStorage and CommonsHttpClient 3.x based RemoteRepositoryStorage.

22.12 Repository Customization

Interface: org.sonatype.nexus.plugins.RepositoryCustomizer

This extension component will be invoked during configuration of every Repository instance, and may be used to add some "extra" configuration to repositories. The procurement plugin uses this mechanism to "inject" RequestProcessor that will evaluate rules before allowing execution of request.

22.13 Item and File Inspectors

Interface: org.sonatype.nexus.proxy.attributes.StorageItemInspector

Interface: org.sonatype.nexus.proxy.attributes.StorageFileItemInspector

Attribute storage ItemInspectors are able to "decorate" items in repositories with custom attributes. Every file stored/cached/uploaded in Nexus will be sent to these components for inspection and potentially decoration. The StorageItemInspector will get all item types for inspection (file, collections, links), while StorageFileItemInspector will only get file items. Currently only one ItemInspector is used in Nexus: the checksumming inspector, that decorates all file items in Nexus with SHA1 checksum and stores it into item attributes.

22.14 Nexus Feeds

Interface: org.sonatype.nexus.rest.feeds.sources.FeedSource

To add new RSS feeds, a plugin may provide implementation of this extension point. Nexus provides implementation for all the "core" RSS feeds.

22.15 Nexus Tasks and Task Configuration

Interface: org.sonatype.nexus.scheduling.NexusTask<T>

Interface: org.sonatype.nexus.tasks.descriptors.ScheduledTaskDescriptor

NexusTask is an extension point to implement new Nexus Scheduled Tasks.

If a contributed task needs UI, then the plugin which provides the NexusTask should provide a ScheduledTaskDescriptor which allows the UI customization for the task creation and management interface.

22.16 Application Customization

Interface: org.sonatype.nexus.rest.NexusApplicationCustomizer

This extension component is able to intercept URLs routed in the Nexus REST API layer.

22.17 Request Processing

Interface: org.sonatype.nexus.proxy.repository.RequestProcessor

This extension point can affect how a repository reacts to an item request.

22.18 Using the Nexus Plugin Archetype

To create a new Nexus Plugin, you can use the Nexus Plugin Archetype by running archetype:generate with the following identifiers:

archetypeGroupId

org.sonatype.nexus.archetypes

archetypeArtifactId

nexus-plugin-archetype

archetypeVersion

1.2

Once you run archetype:generate, you will be prompted for the groupId, artifactId, version, and package name for the generated project.

```
$ mvn archetype:generate -DarchetypeGroupId=org.sonatype.nexus.archetypes ←
  \
-DarchetypeArtifactId=nexus-plugin-archetype \
-DarchetypeVersion=1.2
[INFO] Scanning for projects...
```

```
[INFO]
[INFO]  ↵
----- ↵
[INFO] Building Maven Stub Project (No POM) 1
[INFO]  ↵
----- ↵
[INFO]
[INFO] >>> maven-archetype-plugin:2.1:generate (default-cli) @ standalone- ↵
      pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:2.1:generate (default-cli) @ standalone- ↵
      pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:2.1:generate (default-cli) @ standalone- ↵
      pom ---
[INFO] Generating project in Interactive mode
[INFO] Archetype repository missing. Using the one from [org.sonatype. ↵
      nexus.archetypes:nexus-plugin-archetype:1.2] found in catalog remote
Define value for property 'groupId': : org.sonatype.book.nexus
Define value for property 'artifactId': : sample-plugin
Define value for property 'version': 1.0-SNAPSHOT: :
Define value for property 'package': org.sonatype.book.nexus: :
[INFO] Using property: nexusVersion = 1.4.1
Confirm properties configuration:
groupId: org.sonatype.book.nexus
artifactId: sample-plugin
version: 1.0-SNAPSHOT
package: org.sonatype.book.nexus
nexusVersion: 1.4.1
Y: :
[INFO]  ↵
----- ↵
[INFO] Using following parameters for creating project from Archetype:  ↵
      nexus-plugin-archetype:1.2
[INFO]  ↵
----- ↵
[INFO] Parameter: groupId, Value: org.sonatype.book.nexus
[INFO] Parameter: artifactId, Value: sample-plugin
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: org.sonatype.book.nexus
[INFO] Parameter: packageInPathFormat, Value: org/sonatype/book/nexus
[INFO] Parameter: package, Value: org.sonatype.book.nexus
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: nexusVersion, Value: 1.4.1
[INFO] Parameter: groupId, Value: org.sonatype.book.nexus
```

```
[INFO] Parameter: artifactId, Value: sample-plugin
[INFO] project created from Archetype in dir: /Users/anders/dev/sample- ↵
  plugin
[INFO] ↵
-----
[INFO] BUILD SUCCESS
[INFO] ↵
-----
[INFO] Total time: 54.206s
[INFO] Finished at: Thu Oct 27 11:12:13 CEST 2011
[INFO] Final Memory: 7M/265M
[INFO] ↵
----- ↵
```

Once the Archetype plugin has created the project, you will have a project with the layout shown in Figure 22.1.

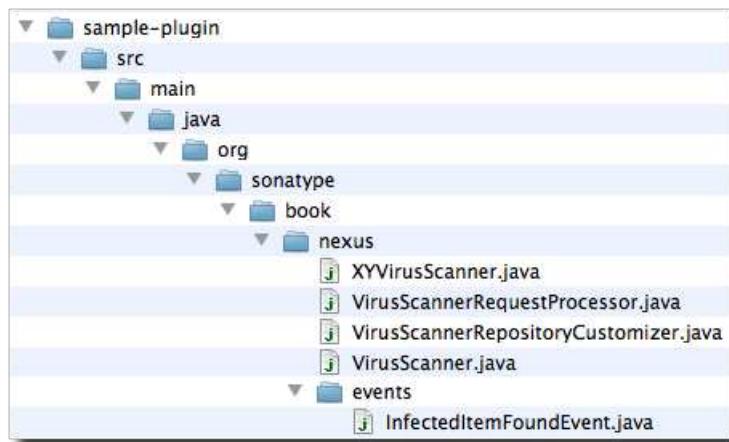


Figure 22.1: Layout of a Nexus Plugin Project

The generated project contains the following classes:

org.sonatype.book.nexus.XYVirusScanner

An implementation of the VirusScanner Interface

org.sonatype.book.nexus.VirusScannerRequestProcessor

A class which customizes the request handling process in Nexus

org.sonatype.book.nexus.VirusScannerRepositoryCustomizer

A class which customizes the repositories affected by the VirusScanner

org.sonatype.book.nexus.VirusScanner

A VirusScanner interface described in Section [22.21](#).

org.sonatype.book.nexus.events.InfectedItemFoundEvent

A simple event which is fired by the VirusScanner

22.19 Set the Target Nexus Version

When creating a new Nexus Plugin project with the Nexus Plugin archetype, a default target Nexus version will be set. To change the target Nexus version of the new Nexus Plugin project, you will need to edit the project's POM.

Open up the POM of your Nexus Plugin and find the section that defines properties. Here you can change the value of the nexus-version property to target a specific Nexus version.

```
<project>
  ...
  <properties>
    <!-- Set the Nexus version here, against which you build the plugin -->
    <nexus-version>1.4.1</nexus-version>
  </properties>
  ...
</project>
```

It is also possible to override the archetype's default value of this property during the creation of a new project. To do that, you would set the "nexusVersion" system property value on the command-line:

```
$ mvn archetype:generate -DarchetypeGroupId=org.sonatype.nexus.archetypes \
  \
-DarchetypeArtifactId=nexus-plugin-archetype \
-DarchetypeVersion=1.2 \
-DnexusVersion=1.9.2
```

22.20 Building a Nexus Plugin Project

To build your Nexus plugin project, just run mvn install in the newly generated project directory. Once the build is completed, your plugin's JAR will be available in the project's target/ folder.

The Nexus Plugin project, as created by the Nexus Plugin archetype, depends on a number of artifacts which may not be available from the Maven Central repository. If you experience missing artifacts during your Nexus plugin project build, you should make sure that the following repositories are added to your environment:

- The Sonatype Forge Repository: <http://repository.sonatype.org/content/groups/forge>
- The Codehaus Snapshot Repository: <http://snapshots.repository.codehaus.org>

If you experience any difficulty with the build, you may also want to remove the section of the generated POM that is labelled with the #ITSet label.

If you are using Nexus, and you have configured your build to work against a public group, you will want to make sure that you have added both of the repositories listed above to your public group. You will also want to make sure that the Repository Policy setting for the Sonatype Forge Repository is set to "Release".

22.21 Creating a Complex Plugin

In this section, we will step through a skeletal, sample project that implements a virus scanner plugin for Nexus. This plugin will consist of:

- A managed "virus scanner" component
- A RequestProcessor that sends all "incoming" artifacts for scanning
- A repository customizer to inject a RequestProcessor to all proxy repositories

We start with creating a @Managed component contract for the VirusScanner. While this class could just as easily be a non-managed component, this example uses the @Managed and @Singleton annotations to demonstrate dependency injection.

VirusScanner Interface

```
package org.sonatype.book.nexus;

import javax.inject.Singleton;

import org.sonatype.nexus.proxy.item.StorageFileItem;
import org.sonatype.plugin.Managed;

@Managed
@Singleton
public interface VirusScanner
{
    boolean hasVirus( StorageFileItem file );
}
```

Once we have the interface for VirusScanner, we need to define a named instance XYVirusScanner which implements the interface. The following example shows how the @Named annotation is used to assign a name of "XY" to this implementation.

XYVirusScanner Implementation

```
package org.sonatype.book.nexus;

import javax.inject.Named;

import org.sonatype.nexus.proxy.item.StorageFileItem;

@Named( "XY" )
public class XYVirusScanner implements VirusScanner {

    public boolean hasVirus( StorageFileItem file ) {

        // DO THE JOB HERE
        System.out.println( "Kung Fu VirusScanner --- " +
            "scanning for viruses on item: " + file.getPath() );

        // simulating virus hit by having the filename
        // contain the "infected" string
        return file.getName().contains( "infected" );
    }
}
```

The next class is a request processor which virus scans an artifact before it is cached locally. If a virus is found in an artifact, this plugin will refuse to cache the artifact and trigger an event which will signal

that a virus was found in a file item. Note the use of @Named which assigns the name "virusScanner" to this component. Also note the two uses of @Inject. The first use of @Inject will fetch the default implementation of ApplicationEventMulticaster, and the second use of @Inject will fetch the "XY" virus scanner.

Virus Scanning Request Processor

```
package org.sonatype.book.nexus;

import javax.inject.Inject;
import javax.inject.Named;

import org.sonatype.book.nexus.events.InfectedItemFoundEvent;
import org.sonatype.nexus.proxy.ResourceStoreRequest;
import org.sonatype.nexus.proxy.access.Action;
import org.sonatype.nexus.proxy.item.AbstractStorageItem;
import org.sonatype.nexus.proxy.item.StorageFileItem;
import org.sonatype.nexus.proxy.repository.ProxyRepository;
import org.sonatype.nexus.proxy.repository.Repository;
import org.sonatype.nexus.proxy.repository.RequestProcessor;
import org.sonatype.plexus.appevents.ApplicationEventMulticaster;

@Named( "virusScanner" )
public class VirusScannerRequestProcessor
implements RequestProcessor {

    @Inject
    private ApplicationEventMulticaster applicationEventMulticaster;

    @Inject
    private @Named( "XY" )
    VirusScanner virusScanner;

    // @Inject
    // private @Named("A") CommonDependency commonDependency;

    public boolean process( Repository repository,
                           ResourceStoreRequest request, Action action )
    {
        // Check dependency
        // System.out.println( "VirusScannerRequestProcessor " +
        //         "--- CommonDependency data: " +
        //         commonDependency.getData()
        // );

        // don't decide until have content
        return true;
    }
}
```

```
public boolean shouldProxy( ProxyRepository repository,
                           ResourceStoreRequest request )
{
    // don't decide until have content
    return true;
}

public boolean shouldCache( ProxyRepository repository,
                           AbstractStorageItem item )
{
    if ( item instanceof StorageFileItem ) {
        StorageFileItem file = (StorageFileItem) item;

        // do a virus scan
        boolean hasVirus = virusScanner.hasVirus( file );

        if ( hasVirus ) {
            applicationEventMulticaster
                .notifyEventListeners(
                    new InfectedItemFoundEvent( item
                        .getRepositoryItemUid().getRepository(),
                        file ) );
        }
    }

    return !hasVirus;
} else {
    return true;
}
}
```

The last extension is RepositoryCustomizer, that simply injects our virus scanner RequestProcessor into proxy repositories only (the only way to get an artifact into Nexus from uncontrolled Internet. Local uploads are considered "safe" for this example). Note how the request processor is injected into this repository customizer with @Inject and @Named.

The Virus Scanner Repository Customizer

```
package org.sonatype.book.nexus;

import javax.inject.Inject;
import javax.inject.Named;

import org.sonatype.configuration.ConfigurationException;
import org.sonatype.nexus.plugins.RepositoryCustomizer;
import org.sonatype.nexus.proxy.repository.ProxyRepository;
```

```
import org.sonatype.nexus.proxy.repository.Repository;
import org.sonatype.nexus.proxy.repository.RequestProcessor;

public class VirusScannerRepositoryCustomizer
    implements RepositoryCustomizer {

    @Inject
    private @Named( "virusScanner" )
    RequestProcessor virusScannerRequestProcessor;

    public boolean isHandledRepository( Repository repository ) {
        // handle proxy repos only
        return repository.getRepositoryKind()
            .isFacetAvailable( ProxyRepository.class );
    }

    public void configureRepository( Repository repository )
        throws ConfigurationException {
        repository.getRequestProcessors()
            .put( "virusScanner",
                virusScannerRequestProcessor );
    }

}
```

22.22 Nexus Plugin Descriptor Maven Plugin

Nexus plugins have a custom packaging "nexus-plugin" which is introduced by the NexusPD Maven Plugin. A "nexus-plugin" packaged plugin:

- is a plain JAR
- has a META-INF/nexus/plugin.xml embedded Nexus Plugin Metadata embedded
- has static resources embedded into the plugin JAR

The NexusPD Maven plugin introduces a new project path (i.e. src/main/static-resources). Static resources such as JavaScript files, images, and CSS which are referenced in

22.23 The Nexus Plugin Descriptor

Every Nexus plugin has a plugin descriptor which is generated during the build process for a plugin. This plugin descriptor is packaged with the plugin JAR and can be found in \$basedir/target/classes/META-INF/nexus/plugin.xml

A Nexus Plugin Descriptor

```
<plugin>
  <modelVersion>1.0.0</modelVersion>
  <groupId>org.sonatype.sample</groupId>
  <artifactId>sample-plugin</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>Nexus Plugin Archetype</name>
  <applicationId>nexus</applicationId>
  <applicationEdition>OSS</applicationEdition>
  <applicationMinVersion>1.4.0</applicationMinVersion>
</plugin>
```

If your Nexus plugin has any dependencies, they will be included in this plugin descriptor automatically. For example, if the Nexus plugin you were developing had a dependency on commons-beanutils version 1.8.2, your plugin descriptor will include the following classpathDependency

```
<plugin>
  <modelVersion>1.0.0</modelVersion>
  <groupId>org.sonatype.book.nexus</groupId>
  <artifactId>sample-plugin</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>Nexus Plugin Archetype</name>
  <applicationId>nexus</applicationId>
  <applicationEdition>OSS</applicationEdition>
  <applicationMinVersion>1.4.0</applicationMinVersion>
  <classpathDependencies>
    <classpathDependency>
      <groupId>commons-beanutils</groupId>
      <artifactId>commons-beanutils</artifactId>
      <version>1.8.2</version>
      <type>jar</type>
    </classpathDependency>
  </classpathDependencies>
</plugin>
```

22.24 Defining Custom Repository Types

When you need to introduce a custom repository type, you should implement the Repository interface. The following example extends the HostedRepository class and adds a repository type with the path prefix "sample".

Creating a Custom Repository Type Interface

```
package org.sample.plugin;

import org.sonatype.nexus.plugins.RepositoryType;
import org.sonatype.nexus.proxy.repository.HostedRepository;

@RepositoryType( pathPrefix="sample" )
public interface SampleRepository extends HostedRepository {
    String boo();
}
```

If you want to implement a custom repository type, you should reference the nexus-proxy module as dependency which contains the AbstractRepository class which is a useful super-class for repository implementations. To implement the SampleRepository interface, you can then extend the AbstractRepository as shown in the following example.

Creating a Custom Repository Type Implementation

```
package org.sample.plugin;

public class DefaultSampleRepository extends AbstractRepository
    implements SampleRepository {
    .... implement it
}
```

Your newly introduced repository type will appear under <http://localhost:8081/nexus/content/sample/>.

Appendix A

Migrating to Nexus from Artifactory

A.1 Introduction

This appendix walks you through the process of using the Nexus Migration Plugin to migrate an existing Artifactory installation to a new Nexus installation. The Nexus Migration Plugin can read an Artifactory System Export archive and recreate groups and repositories based on your existing Artifactory installation. Once the migration is completed, the Nexus Artifactory Bridge will then serve repository artifacts to clients using a web application, exposing the same artifacts to clients using the same URLs and repository identifier that they referenced in your legacy Artifactory installation. This Nexus Migration plugin offers an easy, drop-in replacement for Artifactory.

A.2 Creating an Artifactory Backup

The Nexus Migration Plugin relies on the Artifactory System Export which is a ZIP archive containing all of the configuration and data for an Artifactory installation. To generate this export, log in as an administrative user in Artifactory, and click on the SystemImport & Export in the left navigation menu. Clicking on this link will show the Entire System Export & Import screen as shown in Figure A.1.

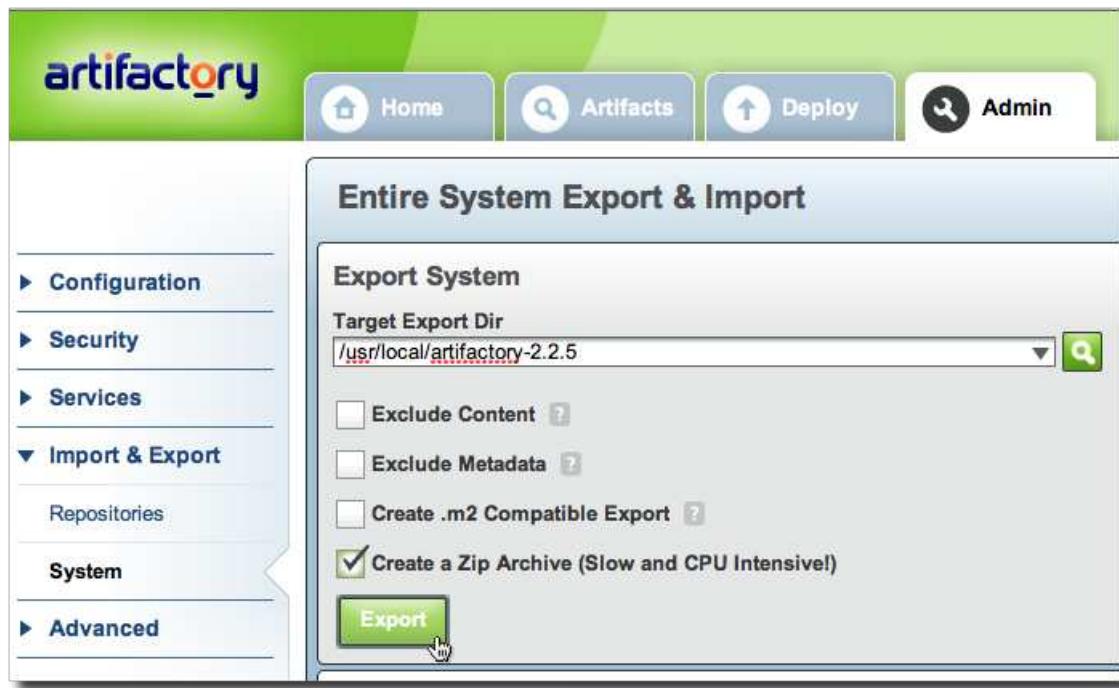


Figure A.1: Creating an Artifactory System Export

This appendix assumes that Artifactory has been installed in the directory `/usr/local/artifactory-2.2.5`. To create a backup ZIP archive in that same directory:

- Enter `/usr/local/artifactory-2.2.5` into the Target export dir field. If you are on a Windows platform, or if you have installed Artifactory in a different path, enter the path at which Artifactory is installed.
- Check the Create a zip archive (slow!) checkbox as shown in Figure A.1.
- Click the Export

When you click the Export button, Artifactory is going to create a timestamped ZIP archive in the specified directory. In this example, Artifactory creates a file named `/usr/local/artifactory-2.2.5/20100222.154349.zip`. Once this step is completed, you will no longer need to run Artifactory, and you can safely shut it down. In the next section you will learn how to install the Nexus Migration Plugin.

A.3 Installation of the Migration Plugin and Artifactory Bridge

The migration plugin is a free, open-source plugin that can be downloaded by visiting <http://nexus.sonatype.org/downloads/> and downloading `nexus-migration-plugin-packaging-1.5-webapp.zip`.

Note

This section assumes that you have just downloaded and installed Nexus Open Source or Nexus Professional. If you haven't installed Nexus, you should make a bookmark for this section and jump to Chapter 3. Once you have installed Nexus, return to this section and install the Nexus Migration Plugin.

The following screen listing shows a series of commands which can be used to install the Nexus Migration plugin into an existing Nexus installation. This listing assumes that the environment variable `$NEXUS_HOME` points to an existing Nexus installation. Download the Nexus Migration plugin, copy the distribution archive to the Nexus installation folder, and unpack it. This archive will install the necessary applications and plugins to enable a smooth migration from Artifactory to Nexus.

```
$ wget http://nexus.sonatype.org/downloads/\
nexus-migration-plugin-packaging-1.5-webapp.zip

...
Resolving repository.sonatype.org... 63.246.20.88
Connecting to repository.sonatype.org|63.246.20.88|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 280179 (274K)
Saving to: 'nexus-migration-plugin-packaging-1.5-webapp.zip'

100%[=====] 280,179      406K/s   in 0.7 ←
 s

2009-03-14 15:03:52 (406 KB/s) - \
'nexus-migration-plugin-packaging-1.5-webapp.zip' saved [280179/280179]
$ cp nexus-migration-plugin-packaging-1.5-webapp.zip ${NEXUS_HOME}
$ cd ${NEXUS_HOME}
$ unzip nexus-migration-plugin-packaging-1.5-webapp.zip
Archive: nexus-migration-plugin-packaging-1.5-webapp.zip
creating: runtime/
creating: runtime/apps/
creating: runtime/apps/artifactory-bridge/
creating: runtime/apps/artifactory-bridge/webapp/
creating: runtime/apps/artifactory-bridge/webapp/WEB-INF/
inflating: runtime/apps/artifactory-bridge/webapp/WEB-INF/web.xml
creating: conf/
```

```
inflating: conf/jetty.xml
creating: runtime/apps/nexus/
creating: runtime/apps/nexus/plugin-repository/
creating: .../nexus-migration-plugin-artifactory-1.2/
...
creating: runtime/apps/nexus/lib/
inflating: .../nexus-migration-plugin-configuration-1.2.jar
inflating: .../nexus-migration-plugin-artifactory-bridge-1.2.jar
```

Warning

The Nexus Migration plugin archive contains a copy of `conf/jetty.xml`. If you have customized `jetty.xml` you will want to make a backup of your customized `jetty.xml` before you install this Migration plugin. Most people using the Nexus Migration plugin are going to be setting up a new instance of Nexus so this shouldn't be an issue.

Once the Nexus Migration plugin has been installed, start up Nexus and go to the Nexus URL. If you are just setting up Nexus, and you've skipped the installation chapter, the default Nexus URL is <http://localhost:8081/nexus> and the default administrative username/password combination is "admin" and "admin123". If the Nexus Migration Plugin is installed properly, you should see the Artifactory Import link under Administration in the Nexus application menu as shown in Figure A.2.

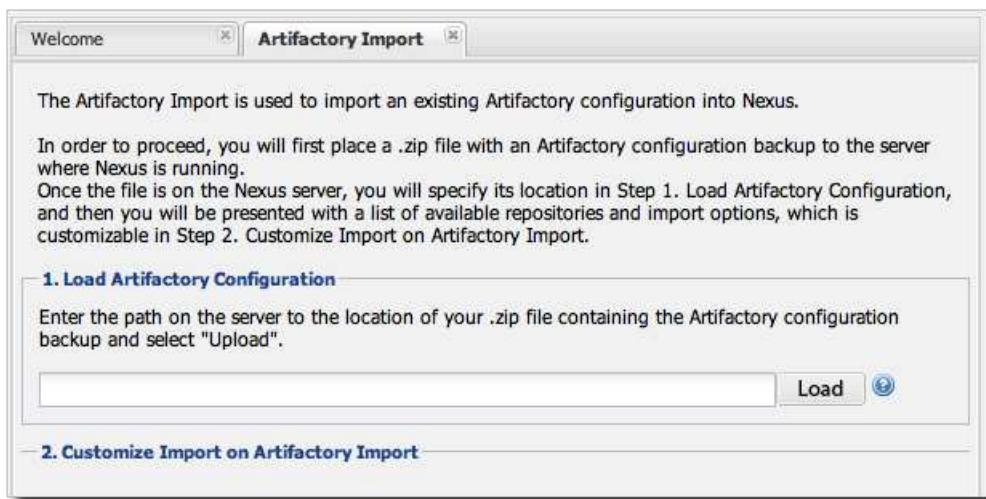


Figure A.2: Artifactory Import Panel from the Nexus Migration Plugin

A.4 Importing an Artifactory System Backup

To import the backup created in Section A.2, enter the path to the Artifactory system backup archive in the Artifactory Import panel as shown in Figure A.3. This path references a filesystem path on the same server that is running Nexus.

Note

Please note that the filesystem path in Step 1 of the Artifactory Import process references a path on the same server that is running Nexus. This field is not a file upload field, and you are not uploading the Artifactory System Backup archive from your local machine to the server. The Artifactory system export archive can be rather large, and the assumption of the Nexus Migration plugin is that you are installing Nexus on the same server that had previously run Artifactory. If you are not installing Nexus on the same server, you must upload the Artifactory System Backup ZIP file to the Nexus server prior to typing in the filesystem path shown in Figure A.3.

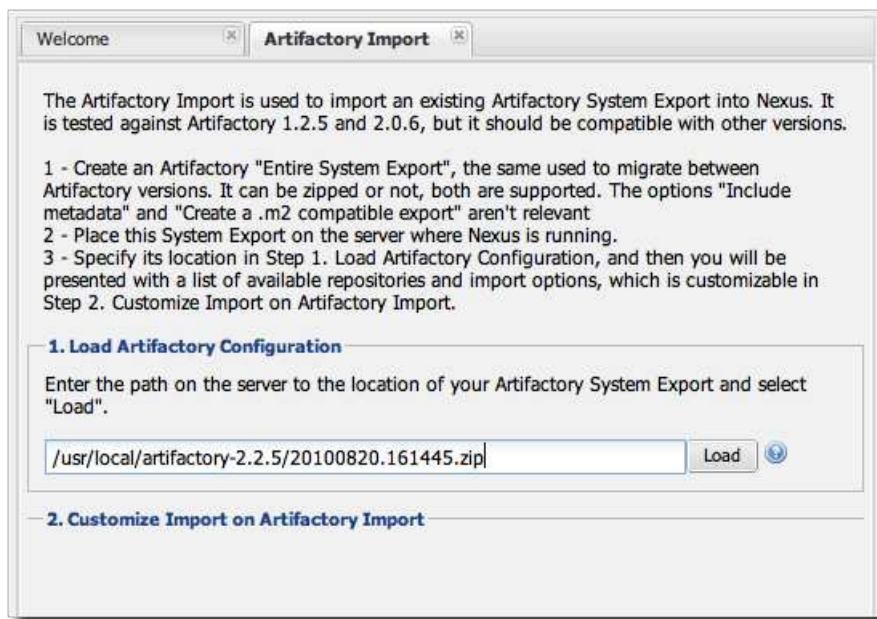


Figure A.3: Specifying the Artifactory System Backup Path

Click the Import button, and Nexus will then examine the Artifactory System Backup archive and populate fields in Step 2 of the Artifactory Import panel.

A.5 Configuring the Artifactory Import

Once the System Backup has been imported, the Nexus Migration plugin will populate the Groups, Repositories, and Users tables in Step 2 of the Artifactory Import panel as shown in Figure A.4. These tables and form elements allow you to customize which repositories and groups are going to be imported into Nexus and how Nexus will configure corresponding repositories. The following sections describe each of these configuration tables in detail.

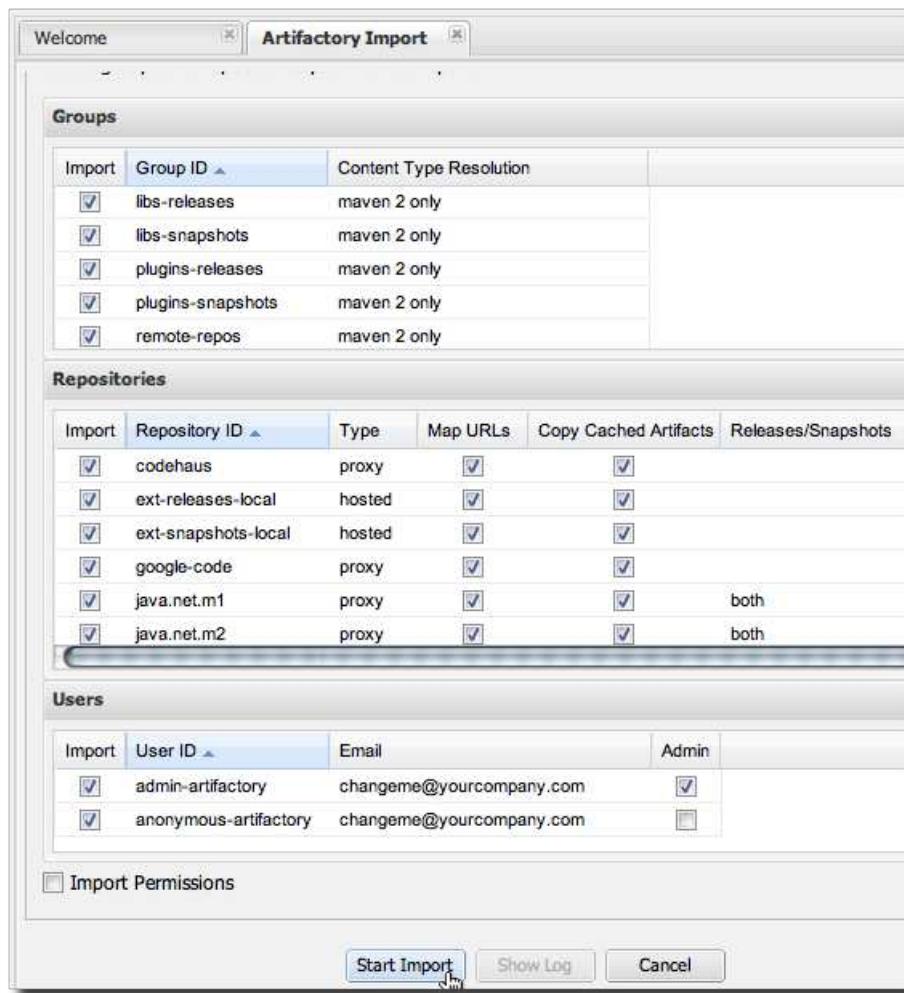


Figure A.4: Configuring the Artifactory Import

A.6 Configuring Artifactory Group Imports

The groups table displays all of the groups contained in the Artifactory system backup. By default, all groups are selected. If you wish to omit an Artifactory group from the import process, deselect the checkbox for that group in the Import column. Each group's content type resolution is shown in the

Content Type Resolution column. Changing the content type resolution will control how Nexus configures the Nexus Group that corresponds to each Artifactory Group.

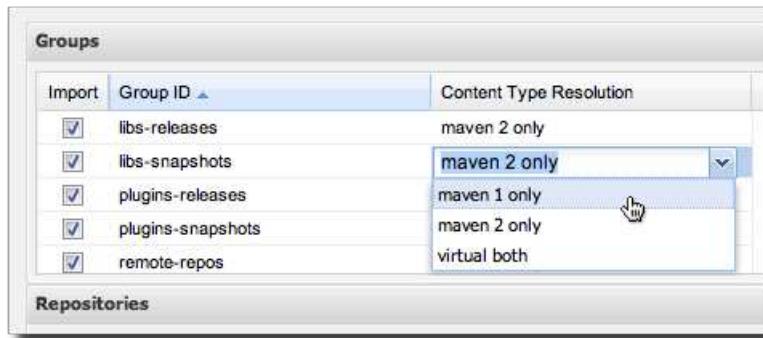


Figure A.5: Configuring Artifactory Group Imports

A.7 Configuring Artifactory Repository Imports

The second table in Step 2 of the Artifactory Import panel is the Repositories panel. In this panel you can select repositories to import and also configure how Artifactory proxy repositories would be mapped to Nexus proxy repository with matching canonical URLs. Each row in the Repositories table contains the following columns:

Import

This column contains a simple checkbox. If the checkbox is checked, the repository will be imported into Nexus. If the checkbox is not checked, the Nexus Migration plugin will omit the repository from the import process.

Repository ID

This is the repository identifier from Artifactory. This column cannot be edited.

Type

This is the type of the repository being imported. Hosted repositories in Artifactory will be imported as hosted repositories in Nexus. Proxy repositories in Artifactory will be imported as proxy repositories in Nexus unless they are merged with an existing proxy repository.

Map URLs

Since Nexus is a drop-in replacement for Artifactory, the Nexus Migration Plugin gives you the option to Map the legacy Artifactory URLs to the newer Nexus URLs. Checking this box will tell

the Nexus Migration plugin to create the appropriate URL mappings to allow clients to interact with the newly created Nexus repositories via the legacy Artifactory URLs.

Copy Cached Artifacts

Checking this box will tell the Nexus Migration plugin to copy the cache contents of a proxy repository to the newly created Nexus repository. Leaving this box unchecked will tell the Nexus Migration plugin to only copy the configuration for a proxy repository.

Releases/Snapshots

This allows you to customize the Repository Policy in a newly created Nexus repository. If you want a hosted or proxy repository to serve only release or snapshot versions, select the corresponding value in the drop-down. If you want a host or proxy repository to serve both release and snapshot versions, select both in this drop-down list. If you want Nexus to use the type implied by the Artifactory repository, don't select a value in this drop-down.

Merge With

If an Artifactory proxy repository has a canonical URL which matches the canonical URL of an existing Nexus repository, you can merge this proxy repository into the existing Nexus proxy repository. Most people running the Artifactory Import process will likely have a repo1 proxy repository in Artifactory for the Maven Central repository and a central proxy repository in Nexus for the Maven Central repository. As both of these repositories have the same URL, the Nexus Migration plugin will show a checkbox in the Merge With column as shown in Figure A.6. Checking this box will instruct the Nexus Migration plugin to merge the contents and configuration of the repo1 repository with the existing central repository. In other words, if this box is checked, the Nexus Migration plugin will not create a new proxy repository for repo1, and if the box isn't checked, the Nexus Migration plugin will create new proxy repository for repo1

Import	Repository ID	Type	Map URLs	Copy Cached Artifacts	Merge With
<input checked="" type="checkbox"/>	jboss	proxy	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input checked="" type="checkbox"/>	libs-releases-local	hosted	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input checked="" type="checkbox"/>	libs-snapshots-local	hosted	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input checked="" type="checkbox"/>	plugins-releases-local	hosted	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input checked="" type="checkbox"/>	plugins-snapshots-local	hosted	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input checked="" type="checkbox"/>	repo1	proxy	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> central

Figure A.6: Merging a Proxy Repository During Artifactory Import

A.8 Configuring Users and Privileges in the Artifactory Import

The Users table shown in Figure A.7 allows you to select which users are imported into Nexus from the Artifactory system backup archive. The columns in the User table are:

Import

If the checkbox in this column is selected, the Artifactory user will be imported into Nexus, if the box is not selected this user will not be imported into Nexus.

User ID

This is the User identifier from Artifactory, this field cannot be edited.

Email

This is the email of a User, this field cannot be edited.

Admin

If this checkbox is selected, Nexus will associate the newly imported user with the Nexus Administrator role.

Import	User ID	Email	Admin
<input checked="" type="checkbox"/>	admin-artifactory	changeme@yourcompany.com	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	anonymous-artifactory	changeme@yourcompany.com	<input type="checkbox"/>
<input checked="" type="checkbox"/>	deploy	deploy@company.com	<input type="checkbox"/>

Import Permissions

Figure A.7: Configuring Users During Artifactory Import

Just below the Users table is the Import Permissions checkbox. If this checkbox is selected, Nexus will attempt to import Artifactory privileges into Nexus and map the corresponding roles and privileges to the imported users.

A.9 Performing the Artifactory Import

Once the Artifactory import has been configured, click on the Start Import button at the bottom of the Artifactory Import panel. Clicking on the Start Import button will show you the warning dialog shown in Figure A.8.



Figure A.8: Loading Artifactory Configuration Warning

If you are migrating a large Artifactory installation, the loading process could take a significant amount of time. If you are comfortable with your import configuration, click on the Yes button. Clicking on the Yes button will then cause Nexus to display the dialog shown in Figure A.9.

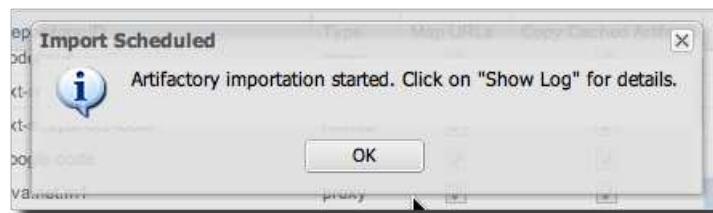
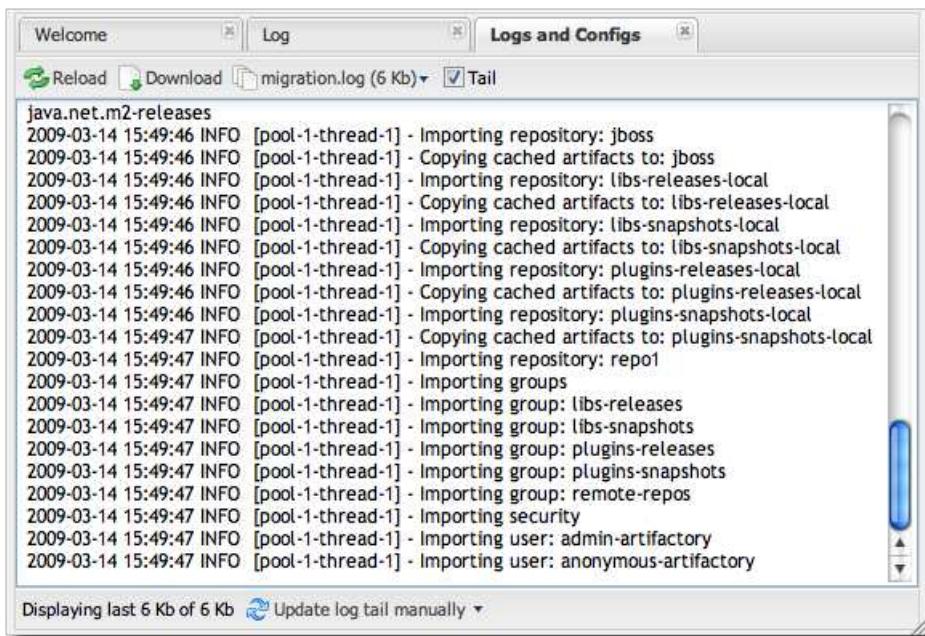


Figure A.9: Artifactory Import Scheduled Dialog

This dialog informs you that the import is scheduled to start as soon as possible using the Nexus scheduled Job execution infrastructure. You can check on the progress of the import process by clicking on the Show Log button at the bottom of the Artifactory Import panel or by clicking on Logs and Config Files under the Views.Repositories menu and selecting the migration.log file from the "Select a document..." drop-down.

The migration log is shown in Figure A.10.



A screenshot of the Nexus 'Logs and Configs' interface. The window title is 'Logs and Configs'. Below it, there are tabs for 'Welcome', 'Log', and 'Logs and Configs'. The 'Log' tab is selected. In the center, there is a scrollable text area titled 'migration.log (6 Kb)'. The log file contains several lines of INFO-level log entries from the 'java.net.m2-releases' logger. The entries describe the import process for various repositories like 'jboss', 'libs-releases-local', 'libs-snapshots-local', 'plugins-releases-local', 'plugins-snapshots-local', and 'repo1'. It also shows imports for groups such as 'libs-releases', 'libs-snapshots', 'plugins-releases', 'plugins-snapshots', and 'remote-repos'. Security and user imports for 'admin-artifactory' and 'anonymous-artifactory' are also listed. At the bottom of the log area, there is a message 'Displaying last 6 Kb of 6 Kb' and a link 'Update log tail manually'.

```
java.net.m2-releases
2009-03-14 15:49:46 INFO [pool-1-thread-1] - Importing repository: jboss
2009-03-14 15:49:46 INFO [pool-1-thread-1] - Copying cached artifacts to: jboss
2009-03-14 15:49:46 INFO [pool-1-thread-1] - Importing repository: libs-releases-local
2009-03-14 15:49:46 INFO [pool-1-thread-1] - Copying cached artifacts to: libs-releases-local
2009-03-14 15:49:46 INFO [pool-1-thread-1] - Importing repository: libs-snapshots-local
2009-03-14 15:49:46 INFO [pool-1-thread-1] - Copying cached artifacts to: libs-snapshots-local
2009-03-14 15:49:46 INFO [pool-1-thread-1] - Importing repository: plugins-releases-local
2009-03-14 15:49:46 INFO [pool-1-thread-1] - Copying cached artifacts to: plugins-releases-local
2009-03-14 15:49:46 INFO [pool-1-thread-1] - Importing repository: plugins-snapshots-local
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Copying cached artifacts to: plugins-snapshots-local
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing repository: repo1
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing groups
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing group: libs-releases
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing group: libs-snapshots
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing group: plugins-releases
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing group: plugins-snapshots
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing group: remote-repos
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing security
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing user: admin-artifactory
2009-03-14 15:49:47 INFO [pool-1-thread-1] - Importing user: anonymous-artifactory
```

Figure A.10: Viewing the Migration Logs

A.10 Configuring Artifactory Clients to Use Nexus

Nexus is a drop-in replacement for Artifactory, this means that you don't need to reconfigure your project's settings.xml and repository elements throughout your POMs to start using Nexus. When you installed the Nexus Migration plugin, you also installed a web application called the Artifactory Bridge which should be running under the context path /artifactory next to the /nexus application. If you run Nexus at the default [URL](http://localhost:8081/nexus/), the artifactory bridge will be running at <http://localhost:8081/artifactory/>

This Artifactory Bridge will serve repository metadata and artifacts from Nexus under the same URLs that your existing Artifactory clients expect. When you were configuring your repositories during the Artifactory Import in Section A.7, you'll remember that there was a column named Map URL check box for a repository instructed the Nexus Migration plugin to configure a mapping between the old Artifactory URL and repository identifier and the new Nexus URL and repository identifier.

To see this mapping in action, take a look at the Central repository through three different URLs. This example assumes that you mapped your old repo1 Artifactory proxy repository to the Nexus central proxy repository. Let's take a quick look at the URL for metadata.xml for the nexus-index artifact which is under the group identifier org.sonatype.nexus. To retrieve this directly from the Maven Central repository, you would use the following URL:

```
http://repo1.maven.org/maven2/\
org/sonatype/nexus/nexus-indexer/maven-metadata.xml
```

To retrieve the same file from the Nexus central proxy of the Maven Central repository, you would use the following URL which contains the new Nexus repository identifier "central":

```
http://localhost:8081/nexus/content/repositories/central/\
org/sonatype/nexus/nexus-indexer/maven-metadata.xml
```

To retrieve the same file from the Artifactory Bridge, you would use the following URL which contains the old Artifactory repository identifier "repo1":

```
http://localhost:8081/artifactory/repo1/\
org/sonatype/nexus/nexus-indexer/maven-metadata.xml
```

The configuration for these mappings is stored in sonatype-work/nexus/conf/mapping.xml. This is an XML file which was configured during the import that maps the old Artifactory repository identifiers to the new Nexus repository identifiers. The Artifactory Bridge web application is configured in \$NEXUS_HOME/conf/plexus.xml and contains a reference to the web application and also configures the context-path /artifactory

Appendix B

Migrating to Nexus from Archiva

B.1 Introduction

This appendix walks you through the process of migrating an existing Archiva installation to a new Nexus installation.

B.2 Migrating Archiva Repositories

Archiva uses the filesystem to store hosted repositories and proxied repositories, because of this migrating from Archiva to Nexus couldn't be simpler. The following sections outline the process for migrating existing Archiva repositories to a new Nexus instance.

B.3 Migrating an Archiva Managed Repository

Archiva Managed Repositories are the equivalent of Nexus Hosted repositories. To migrate a Managed Repository from Archiva to Nexus, all you need to do is:

- Create a New Hosted Repository in Nexus
- Copy the Contents of the Archiva Managed Repository to the Storage Directory of the newly-created Nexus Hosted Repository
- Rebuild the Index for the New Nexus Hosted Repository

The following example will walk through the process of migrating the Archiva repository named "internal" to a new Nexus Hosted repository named "internal". To view your managed repositories in Archiva, login to Archiva as an administrative user and click on the "Repositories" link in the left-hand navigation menu. Clicking on "Repositories" will list all of your Archiva Managed repositories as shown in Figure B.1.

The screenshot shows the 'Managed Repositories' page in Archiva. At the top, there is a header with a '+' icon and the word 'Add'. Below the header, the title 'Archiva Managed Internal Repository' is displayed, accompanied by a small orange icon. To the right of the title are three buttons: 'Edit' (with a pencil icon), 'Delete' (with a red X icon), and a feed icon. The main content area contains the following information:

Identifier	internal
Name	Archiva Managed Internal Repository
Directory	./data/repositories/internal
WebDAV URL	http://localhost:8080/archiva/repository/internal/
Type	Maven 2.x Repository
Releases Included	
Snapshots Included	
Scanned	
Scanning Cron	0 0 * * *
Actions	Scan Repository Now
Stats	No Statistics Available.
POM Snippet	Show POM Snippet

Figure B.1: Archiva Managed Repositories

To migrate this Managed repository to a Nexus Hosted repository, you will need to find the directory in which Archiva stores all of the repository artifacts. to do this, click on the Edit link listed next to the name of the repository you want to migrate as shown in Figure B.1. Click on Edit should load the form shown in Figure B.2.

Admin: Edit Managed Repository

ID: internal

*Name**: Archiva Managed Internal Repository

*Directory**: ./data/repositories/internal

Index Directory:

Type: Maven 2.x Repository

*Cron**: 0 0 * * * ?

Repository Purge By Days Older Than: 30

Repository Purge By Retention Count: 2

Releases Included

Snapshots Included

Scannable

Delete Released Snapshots

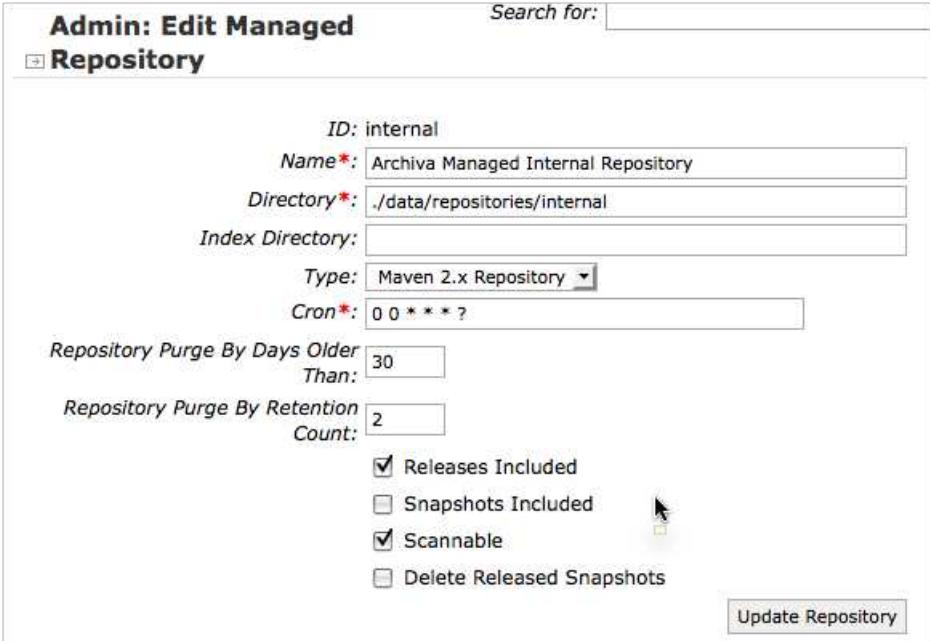


Figure B.2: Editing an Archiva Managed Repository

Take note of the file path for Directory. The file path shown in Figure B.2 is ./data/repositories/internal. If Archiva is installed in /usr/local/archiva-1.2.1, this would correspond to the directory /usr/local/archiva-1.2.1/data/repositories/internal. You will use this path later in this section to copy the contents of your old Archiva Managed Repository to your new Nexus Hosted Repository.

Next, create a new Nexus repository with the same identifier and Name as the old Archiva Managed Repository. To do this, log into Nexus as an administrative user, click on Repositories in the left-hand Nexus navigation menu, and then click on the Add drop-down as shown in Figure B.3. Select "Hosted Repository" and then fill out the Repository ID and Repository Name to match the name of the old Archiva repository. If you are migrating a Snapshot repository, select a Repository Policy of Snapshot, and if you are migrating a Release repository select a Snapshot Policy of Release.

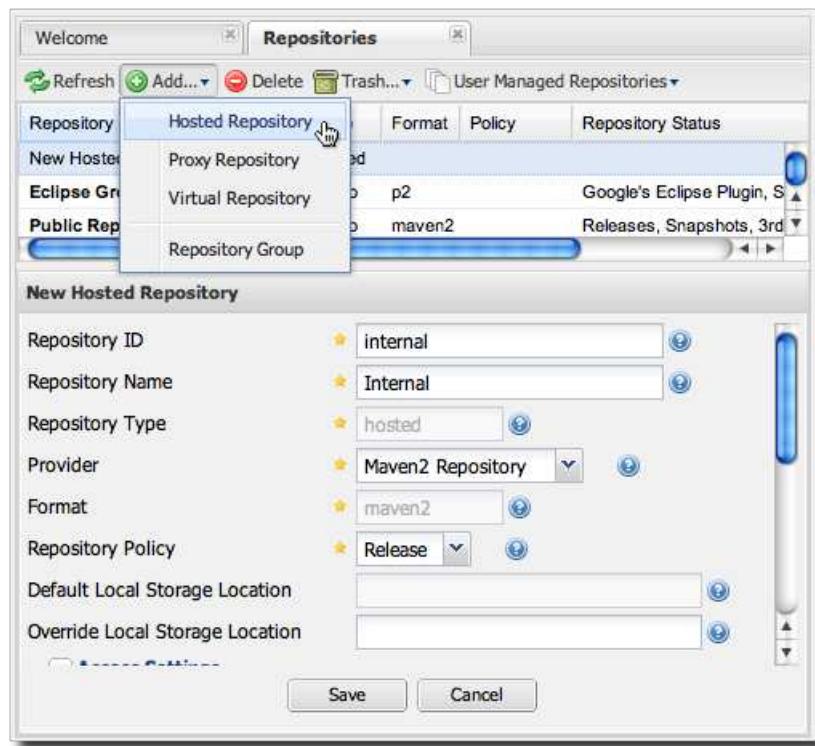


Figure B.3: Creating a Nexus Hosted Repository

Now, you'll need to copy the Archiva repository to the Nexus repository. You can do this, by copying the contents of the Archiva repository directory to the Nexus repository storage directory. If we assume that Archiva is install in /usr/local/archiva-1.2.1, Nexus is install in /usr/local/nexus-1.3.6, and the Sonatype Work directory is /usr/local/sonatype-work. You can copy the contents of the Archiva managed repository to the new Nexus hosted repository by executing the following command:

```
$ cp -r /usr/local/archiva-1.2.1/data/repositories/internal/* \
/usr/local/sonatype-work/nexus/storage/internal/
```

If you are migrating to a Nexus instance which is on a different server, you can simply create an archive of the /usr/local/archiva-1.2.1/data/repositories/internal directory, copy it to the new server, and then decompress your repository archive in the appropriate directory.

Warning

 Archiva stores artifacts from proxied remote repositories in the same directory as artifacts in a managed repository. If you have been proxying a remote repository, you might want to remove artifacts that have been proxied from a remote repository. For example, if your organization uses a groupId of org.company for internal project, you can make sure to only copy the artifacts under the corresponding org/company/

Once the contents of the repository have been copied to the Nexus Hosted repository, you must rebuild the repository index as shown in Figure B.4. Right-clicking on the repository in the list of Nexus repositories will display the context menu shown in the following figure.

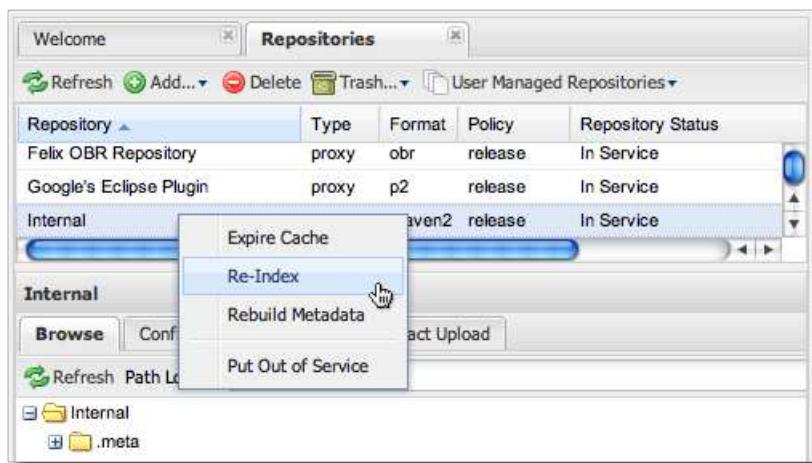


Figure B.4: Rebuilding the Index of a Nexus Hosted Repository

Once the migration is complete, you will be able to search and browse the contents of your newly migrated Nexus Hosted repository.

B.4 Migrating an Archiva Proxy Connector

Archiva allows you to define remote repositories and repository connectors to proxy remote repositories and cache remote artifacts from remote repositories in Archiva Managed Repositories. While Nexus also

provides Proxy repositories, there is one major difference between Nexus and Archiva. Where Nexus maintains a separate local storage directory for each proxy repository, Archiva combines cached remote artifacts into a single filesystem with the contents of a managed repository. In other words, there is no good way to transfer an existing local cache of artifacts between Archiva and Nexus without manually manipulating the contents of Archiva's Managed Repository directory.

To recreate an Archiva repository connector in Nexus as a Proxy repository and to preserve the local cache of artifacts from this repository. You'll need to create a Proxy repository in Nexus, copy the contents of the existing proxy repository to the Nexus storage location for your new Proxy repository, and then rebuild the metadata of your new Nexus Proxy repository.

First step is to take a look at the Remote Repositories in your Archiva installation: log in as an administrative user and then click on "Repositories" under the Administration menu in the left-hand Archiva navigation menu. Once you've clicked this link and loaded the list of repositories, scroll to the bottom of the page to see the list of remote repositories as shown in Figure B.5.

The screenshot shows the 'Remote Repositories' section of the Archiva administration interface. It lists two repositories:

Identifier	Name	URL	Type
central	Central Repository	http://repo1.maven.org/maven2	Maven 2.x Repository
maven2-repository.dev.java.net	Java.net Repository for Maven 2	http://download.java.net/maven/2/	Maven 2.x Repository

Figure B.5: Browsing Archiva Remote Repositories

Defining a proxy repository in Archiva involves associating one of the remote repositories defined in Figure B.5 with one of the Managed Repositories defined in Figure B.1. Once you do this, requests for artifacts from the managed repository will also query the remote repository. If an artifact is found in the remote repository, it will be retrieved and stored in the managed repository's storage directory. To see a list of proxy connectors and the managed repositories they are associated with, click on "Proxy Connectors" in the left-hand Archiva menu, you will see a list similar to that shown in Figure B.6.

The screenshot shows the 'Repository Proxy Connectors' section of the Archiva interface. At the top, there is a header with a 'Add' button. Below it, a section titled 'internal' contains the text 'Archiva Managed Internal Repository' with a small icon. Underneath this, there are two 'Proxy Connector' entries:

- Proxy Connector**: central
Central Repository
<http://repo1.maven.org/maven2>
- Proxy Connector**: maven2-repository.dev.java.net
Java.net Repository for Maven 2
<http://download.java.net/maven/2/>

Each entry has a 'Settings' link below it and a row of icons (Edit, Delete, etc.) to its right.

Figure B.6: Archiva Proxy Connectors

Click on the Edit Icon (or Pencil) next to second Proxy Connector listed in Figure B.6, this will then load the settings form for this proxy connector shown in Figure B.7. You should use the settings for this proxy connect to configure your new Nexus Proxy repository.

The screenshot shows the configuration interface for a Proxy Connector in Archiva. At the top, there are dropdown menus for 'Network Proxy' (set to 'direct connection'), 'Managed Repository' (set to 'internal'), and 'Remote Repository' (set to 'maven2-repository.dev.java.net'). Below these are several policy settings: 'Return error when' (set to 'always'), 'On remote error' (set to 'stop'), 'Releases' (set to 'once'), 'Snapshots' (set to 'never'), 'Checksum' (set to 'fix'), and 'Cache failures' (set to 'yes'). Under the 'Properties' section, there is a field labeled 'Properties:' followed by a text input and a 'Add Property' button. A note below says 'No properties have been set.' In the 'Black List' section, there is a field labeled 'Black List:' followed by a text input and a 'Add Pattern' button. A note below says 'No black list patterns have been set.' In the 'White List' section, there is a field labeled 'White List:' followed by a text input and a 'Add Pattern' button. Below these lists, three specific patterns are listed with red 'X' marks: '"javax/**"', '"org/jvnet/**"', and '"com/sun/**"'. At the bottom right of the dialog is a 'Save Proxy Connector' button.

Figure B.7: Archiva Proxy Connector Settings

To create a Proxy repository that will correspond to the Proxy Connector in Archiva, log into Nexus as an administrative user, and click on Repositories in the left-hand Nexus menu. Once you can see a list of Nexus repositories, click on Add... and select Proxy Repository from the drop-down of repository types. In the New Proxy Repository form (shown in Figure B.8) populate the repository ID, repository Name, and use the remote URL that was displayed in Figure B.5. You will need to create a remote repository for every proxy connector that was defined in Archiva.

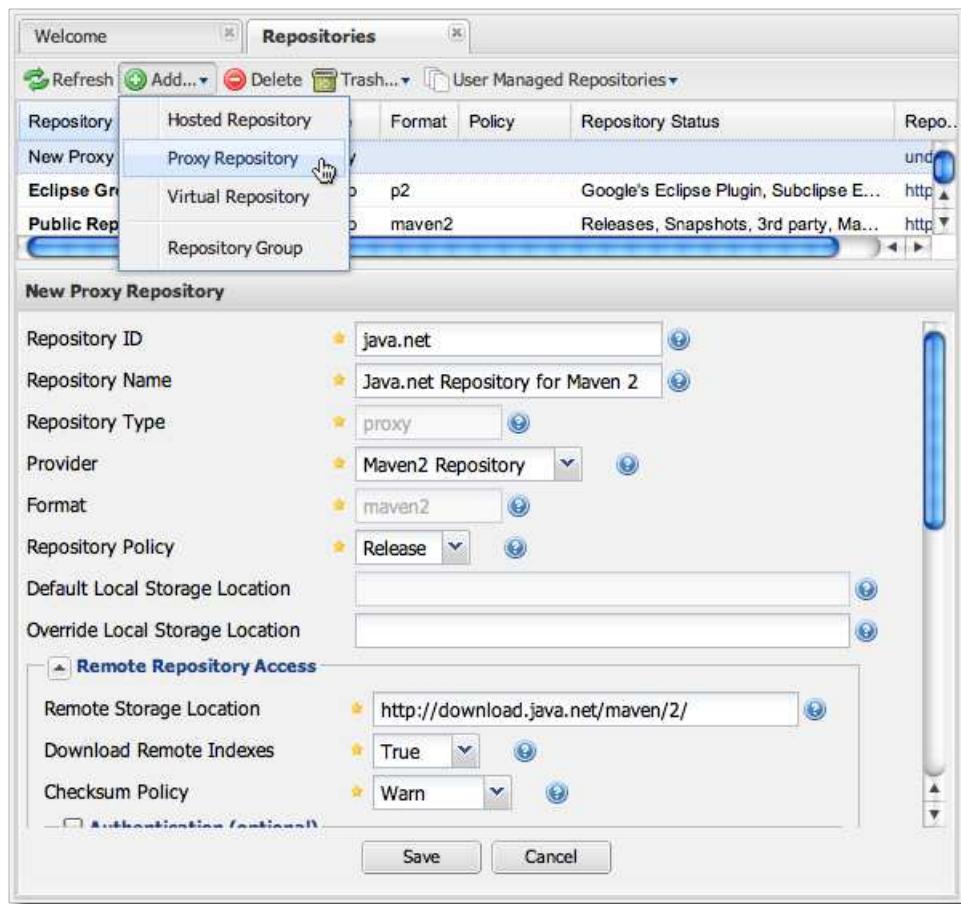


Figure B.8: Creating a Nexus Proxy Repository

To expose this new Proxy repository in a Repository Group, create a new Nexus Repository group or select an existing group by clicking on Repositories in the left-hand Nexus menu. Click on a repository group and then select the Configuration tab to display the form shown in Figure B.9. In the Configuration tab you will see a list of Order Group Repositories and Available Repositories. Click and drag your new Nexus Proxy repository to the list of Ordered Group Repositories, and click Save.

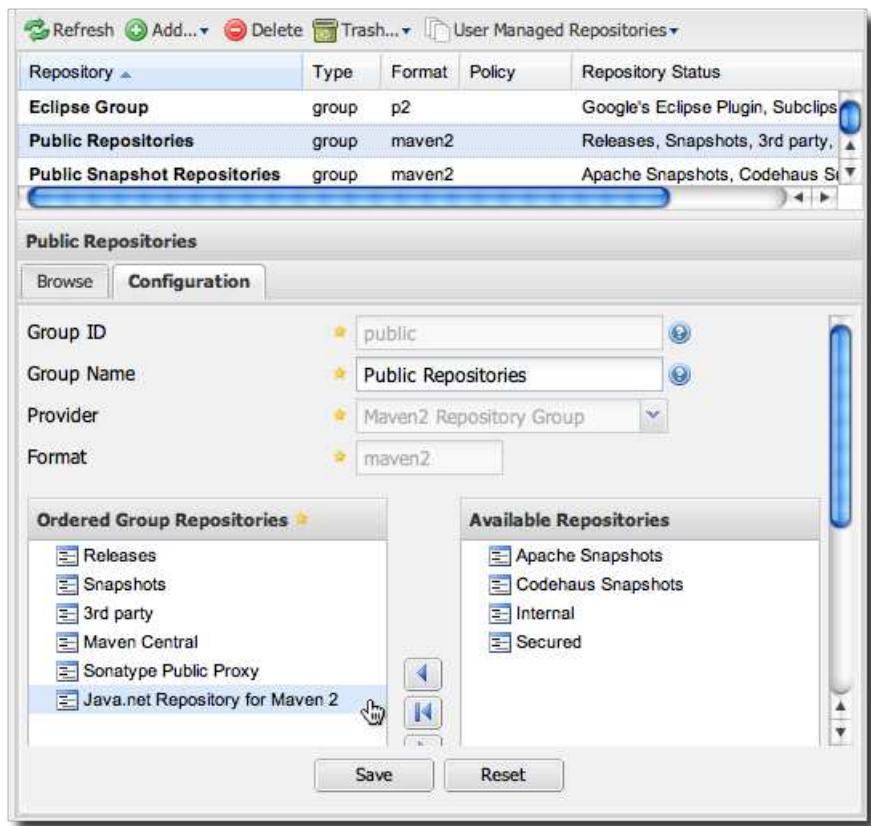


Figure B.9: Adding a Proxy Repository to a Repository Group

Next, you will need to define repository groups that will tell Nexus to only locate certain artifacts in the newly created proxy repository. In , Archiva defined three patterns that were used to filter artifacts available from the proxy connector. These three patterns were "javax/", "com/sun/", and "org/jvnet/**". To recreate this behaviour in Nexus, define three Routes which will be applied to the group you configured in Figure B.9. To create a route, log in as an administrative user, and click on Routes under the Administration menu in the left-hand Nexus menu. Click on Add.. and add three inclusive routes that will apply to the repository group you configured in Figure B.9.

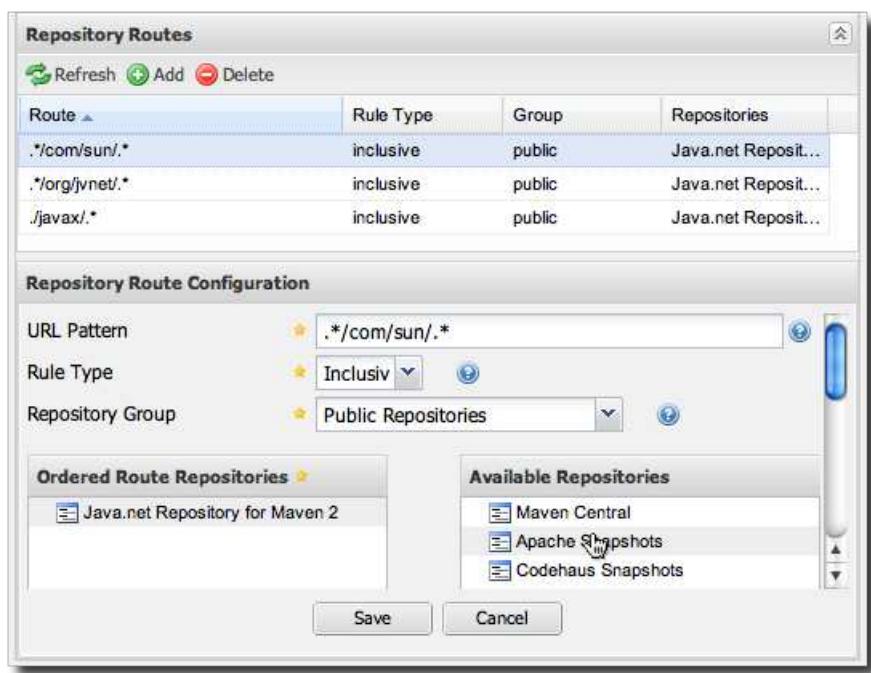


Figure B.10: Defining Nexus Repository Groups

Appendix C

Configuring Nexus for SSL

C.1 Introduction

This chapter contains two sections: the first section details a procedure for connecting Nexus to a remote repository which requires client-side SSL certificates and the second section details the configuration for serving SSL directly from Nexus. When you set up a repository manager for a team of developers spread out over a variety of locations both internal and external to a corporate network, you will likely want to secure your repository using SSL. The instructions in this chapter assume that you are running Nexus embedded in Jetty.

C.2 Importing a SSL Client Certificate

If you need to configure Nexus to proxy a remote repository which requires a SSL Client Certificate, you'll need to import the certificate included with your Nexus license into the JVM used to run your Nexus instance. To make this process simpler, you can use our `import-ssl` tool. We've created a simple command-line utility which automates the process of loading the Server SSL Chain and the client certificate into a JVM.

C.2.1 Downloading the SSL Import Tool

The import-ssl tool can be downloaded from: <http://central.sonatype.com/help/import-ssl.jar>

C.2.2 Importing a Client Certificate

Importing a client certificate involves two steps: importing the server's SSL chain and importing the client SSL key/certificate pair. Some notes about the location of the key-store and default key-store passwords:

- If you are using the default JSSE key-store locations on either a Linux or OS X platform, you must run the commands below as the root user. You can do this either by changing to the root user (`su -`), or by using the `sudo` command: `sudo [command]`.
- The default password used by Java for the built-in key-stores is `changeit`. If your key-store uses a different password, you'll need to specify that password as the last parameter on the command lines above.
- If you want to specify your own key-store location, provide that in place of `<keystore_dir>` in the examples below.
- If you're using a password other than `changeit` for your keystore, you should supply it immediately following the keystore path in the commands below.
- If you specify a keystore location that doesn't exist, the import-ssl utility will create it on-demand.

Before you begin the process of importing a Server SSL Chain and a client certificate you will need three things:

- Network access to the SSL server you are connecting to,
- An SSL client certificate,
- and a certificate password.

C.2.3 Import the Server SSL Chain

The first command imports the entire self-signed SSL certificate chain for `central.sonatype.com` into your JSSE keystore:

```
$ java -jar import-ssl.jar server central.sonatype.com \
<keystore_dir>
```

You would substitute the server name used in the previous listing with the server name you are attempting to connect to. This particular command will connect to <https://central.sonatype.com>, retrieve, and import the server's SSL certificate chain.

C.2.4 Import the Client SSL Key/Certificate Pair

The second command imports your client-side SSL certificate into the JSSE keystore, so Nexus can send it along to the server for authentication:

```
$ java -jar import-ssl.jar client <your-certificate.p12> \
<your-certificate-password> <keystore_dir>
```

When the client command completes, you should see a line containing the keystore path, like the one that follows. This path is important; you will use it in your Nexus configuration below, so make a note of it!

```
...
Writing keystore: /System/Library/Frameworks/JavaVM.framework/\
Versions/1.6.0/Home/lib/security/jssecacerts
```

C.2.5 Configuring Nexus Start-up

Once both sets of SSL certificates are imported to your keystore, you can modify the Nexus \$NEXUS_HOME/conf/wrapper file to inject the JSSE system properties necessary to use these certificates, as seen below.

Note

In the following example, line prefixes like wrapper.java.additional.4 are meant to be appended to the existing wrapper.java.additional.* lines in the wrapper.conf file. In future versions of Nexus, new JVM command-line arguments may be specified in this file. In such a case, where the specific numbers 4 and 5 may be taken, simply increment and use the next two unused numbers.

```
wrapper.java.additional.4=-Djavax.net.ssl.keyStore=<keystore_dir>
```

```
wrapper.java.additional.5=-Djavax.net.ssl.keyStorePassword=< ↵
keystore_password>
```

Once you have configured the Nexus start-up option shown above, restart Nexus and attempt to proxy a remote repository which requires an SSL client certificate. Nexus will use the keystore location and keystore password to configure the SSL interaction to accept the server's SSL certificate and send the appropriate client SSL certificate.

C.3 Configuring Nexus to Serve SSL

If you need to serve repository content using SSL, you can always proxy Nexus with a server like Apache httpd. Apache httpd can easily be configured to serve SSL content using mod_ssl, and there is a large amount of reference material available for configuring httpd to serve secure content. Jetty can also be configured to serve SSL content directly, and if you would like to avoid the extra work of putting a web server like Apache httpd in front of Nexus, this section shows you how to do that. To configure Nexus to serve SSL directly to clients, you'll need to perform the following steps.

Note

All examples given here can be found in the [Nexus source](#), or in the Nexus distribution under `$(NEXUS_HOME)/conf/examples`. Before you customize your Nexus configuration to serve SSL, keep in mind the following:

- Customizations in this Appendix assume that your are running Nexus 1.9.2.
- Any custom Jetty configuration must be contained in the `$(NEXUS_HOME)/conf/jetty.xml` file, or else in the location referenced by the `jetty.xml` property in `$(NEXUS_HOME)/conf/nexus.properties` (in case you've customized this location).
- While the instructions below will work with Nexus Open Source, these instructions assume the filesystem of Nexus Professional. If you are missing Jetty JAR files, you should obtain them from the Jetty project page: <http://www.mortbay.org/jetty/>

C.3.1 Configure the Java Keystore

Follow the instructions on the [How to configure SSL](#) on the Jetty Wiki to setup the appropriate keys and certificates in a form that Jetty can use. Pay particular attention to steps 1-3, and the section at the bottom called Password Issues.

The jetty-util jar and the main Jetty jar can be found in \$NEXUS_HOME/runtime/apps/lib/nexus. The command line used to import an OpenSSL key+cert in PKCS12 format is:

```
$ java -classpath jetty-util-6.1.12.jar:jetty-6.1.12.jar \
org.mortbay.jetty.security.PKCS12Import <pkcs12-file> <keystore>
```

The command line used to generate an obfuscated password hash is:

```
$ java -classpath jetty-util-6.1.12.jar:jetty-6.1.12.jar \
org.mortbay.jetty.security.Password <your-password>
<your-password>
OBF:1t2x1toq1to41t39
MD5:6f1ed002ab5595859014ebf0951522d9
```

The OBF line in the previous output will be used in the jetty.xml three times. You'll need to run the previous command three times to generate the obfuscated hash-codes for three passwords:

- The Key Password
- The Trust Store Password
- The Key Store Password

In the next section, the key store and trust store are the same file, with the same password.

C.3.2 Configure Nexus/Jetty to Use the New Keystore

Note

A jetty.xml with the modifications in this section can be found in \$NEXUS_HOME/conf/examples/jetty-ssl.xml, inside your Nexus distribution.

Modify the `nexus-equivalent jetty.xml`

```
<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.nio.SelectChannelConnector">
      <Set name="host">${application-host}</Set>
      <Set name="port">${application-port}</Set>
    </New>
  </Arg>
</Call>
```

with this:

```
<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.security.SslSelectChannelConnector">
      <Set name="host">${application-host}</Set>
      <Set name="port">${application-port}</Set>
      <Set name="maxIdleTime">30000</Set>
      <Set name="keystore">/etc/ssl/keystore</Set>
      <Set name="truststore">/etc/ssl/keystore</Set>
      <Set name="password">OBF:1v2j1uum1xtv1zej1zer1xtn1uvk1v1v</Set>
      <Set name="keyPassword">OBF:1v2j1uum1xtv1zej1zer1xtn1uvk1v1v</Set>
      <Set name="trustPassword">OBF:1v2j1uum1xtv1zej1zer1xtn1uvk1v1v</Set>
    </New>
  </Arg>
</Call>
```

C.3.3 Modify the application-port for SSL connections

The application-port property, referenced in the configuration above, has a default configuration that many people would more naturally associate with non-SSL connections. You may wish to modify this port to something like 8443, or even 443 (if you have root access from which to start Nexus). To change this property, modify the `$(basedir)/conf/nexus.properties`

Note

You may wish to enable both types of connections, with appropriate rewrite rules between them. Such a configuration is beyond the scope of this section; if you're interested, please refer to the [Jetty Wiki](#) for some information to get you started. Additionally, you may need to add extra port properties to the `nexus.properties` configuration file to accommodate both SSL and non-SSL connections.

C.4 Redirecting Non-SSL Connections to SSL

If you want to make it very easy for people to use your Nexus repository, you will want to configure the automatic redirect from the non-SSL port (default 80) to the SSL port (default 443). When this feature is configured, browsers and clients that attempt to interact with the non-SSL port will be seamlessly redirected to the SSL port. If you do not turn on the automatic redirect to SSL, users who attempt to load the Nexus interface via the default port 80 will see a network error.

If you are proxying your Nexus instance with a web server like Apache httpd, you could configure mod_rewrite to automatically redirect browsers to the SSL port, or you can configure Jetty to perform this redirection. To do this in Jetty you use a custom rewrite rule for Jetty that is bundled with Nexus, inside the plexus-jetty6 library found in \$NEXUS_HOME/runtime/apps/nexus/lib

To enable this feature, configure Jetty to serve SSL directly as demonstrated in Section C.3. After you having configured Jetty to serve SSL directly, open your jetty.xml and replace the existing handler/context-collection declaration with a stand-alone context-collection declaration, by replacing this section:

```
<Set name="handler">
    <New id="Contexts" class="org.mortbay.jetty.handler. ContextHandlerCollection">
        <!-- The following configuration is REQUIRED, and MUST BE FIRST.
            It makes the Plexus container available for use in the Nexus webapp --
        . -->
        <Call name="addLifeCycleListener">
            <Arg>
                <New
                    class="org.sonatype.plexus.jetty.custom. InjectExistingPlexusListener" />
            </Arg>
        </Call>

        <!-- The following configuration disables JSP taglib support,
            the validation of which slows down Jetty's start-up significantly --
        . -->
        <Call name="addLifeCycleListener">
            <Arg>
                <New class="org.sonatype.plexus.jetty.custom. DisableTagLibsListener" />
            </Arg>
        </Call>
    </New>
</Set>
```

with this one:

```
<New id="Contexts" class="org.mortbay.jetty.handler. ←
    ContextHandlerCollection">
    <!-- The following configuration is REQUIRED, and MUST BE FIRST.
        It makes the Plexus container available for use in the Nexus webapp ←
            . -->
<Call name="addLifeCycleListener">
    <Arg>
        <New
            class="org.sonatype.plexus.jetty.custom. ←
                InjectExistingPlexusListener" />
    </Arg>
</Call>

<!-- The following configuration disables JSP taglib support, the
    validation of which slows down Jetty's start-up significantly. -->
<Call name="addLifeCycleListener">
    <Arg>
        <New class="org.sonatype.plexus.jetty.custom.DisableTagLibsListener" ←
            />
    </Arg>
</Call>
</New>
```

Now, configure the rewrite handler for Jetty by adding the following section just above the line with `stopAtShutdown` in it:

```
<Set name="handler">
  <New id="Handlers" class="org.mortbay.jetty.handler.rewrite. ↵
    RewriteHandler">
      <Set name="rules">
        <Array type="org.mortbay.jetty.handler.rewrite.Rule">
          <Item>
            <New id="redirectedHttps"
                class="org.sonatype.plexus.jetty.custom.RedirectToHttpsRule">
              <Set name="httpsPort">${application-port-ssl}</Set>
            </New>
          </Item>
        </Array>
      </Set>
    <Set name="handler">
      <New id="Handlers" class="org.mortbay.jetty.handler. ↵
        HandlerCollection">
        <Set name="handlers">
          <Array type="org.mortbay.jetty.Handler">
            <Item><Ref id="Contexts"/></Item>
            <Item>
              <New id="DefaultHandler"
```

```
        class="org.mortbay.jetty.handler.DefaultHandler"/></Item>
<Item>
    <New id="RequestLog"
        class="org.mortbay.jetty.handler.RequestLogHandler"/></ Item>
</Array>
</Set>
</New>
</Set>
</New>
</Set>
```

Modify \$NEXUS_HOME/conf/nexus.properties and add a new property, application-port-ssl. This will allow you to customize both the SSL and non-SSL ports independently:

```
application-port-ssl=8443
```

Appendix D

Contributing to the Nexus Book

This appendix covers the basics of contributing to the book you are currently reading. This book is an open source project, you can participate in the writing effort if you have an idea for documentation. Sonatype's books are different: they are open writing efforts and we see documentation contributions as having equal value to code contributions. If you are interested in our technology, we'd welcome your contribution.

D.1 Contributor License Agreement (CLA)

In order to contribute to the Nexus book, you will first need to fill out a contributor license agreement. This is a legal agreement between you and Sonatype which ensures that your contributions are not covered by any other legal requirements. Sonatype requires contributors to sign this agreement for all major contributions which are larger than a single section. If your contribution consists of finding and fixing simple typos or suggesting minor changes to the wording or sequence of a particular section, you can contribute these changes via the Sonatype JIRA instance. If your contribution involves direct contribution of a number of sections or chapters you will first need to sign our Contributor License Agreement (CLA).

To download the CLA from the following URL: <http://www.sonatype.org/SonatypeCLA.pdf>

Once you have completed and signed this document, you can fax it to: (650) 472-9197

D.2 Contributors, Authors, and Editors

As with any open source effort, the contributors to the Nexus book are grouped into a simple hierarchy. Sonatype's writing efforts are loosely structured, but we have found it necessary to define some formal categories for contributors.

Reviewers

Many individuals have read the book and taken the time to report typos and bugs. Reviewers are always credited in the Foreword of the book and they make an important contribution to the quality of the book.

Contributors

Contributors are individuals who have contributed one or more sections to the book. Many contributors make a one time contribution to a particular section or collection of sections. Contributors are always credited in the Foreword of the book, and if a contributor sustains a constant level of contribution which adds up to the equivalent of an entire chapter, a contributors name will be added to the list of contributing authors.

Authors

Authors have made a significant contribution to the book equal to the equivalent of one or more chapters. A long-time contributor can also be transitioned to the status of Author at the discretion of an Editor. Authors are often given editorial control over specific chapters or sections of a book, working with Contributors to review, accept, and refine contributions to defined sections of the book.

Editors

An Author can also be an Editor. Each book has at least one editor (and ideally no more than two Editors at any time). On Sonatype Open book projects, Editors are the arbiters of content, they review content submissions and make final decisions about content direction.

For each of these levels, the adjective "Active" can be used if a contributor, author, or editor has been active during the previous 12 months. If you have any questions about contributor status, send any enquiries to book@sonatype.com

D.3 Tools Used to Build and Write this Book

The following tools are used to write this book. If you are interested in contributing to this book, you will need to download the following software:

Asciidoc

A very lightweight markup format with a strong community and a collection of solid tools. For more information, see <http://www.methods.co.nz/asciidoc/>

A Text Editor

Emacs, vi, TextMate, Notepad - it doesn't matter. All you need to contribute to this book is a text editor.

Git

Sonatype stores the source code for all books in GitHub so you will need to download Git. In addition to downloading Git, if you need read/write access to the repository you will also need to sign up for an account on GitHub - <http://www.github.com>

Download the latest version of Git from <http://git-scm.com/>

This next set of tools are optional, and are only required if you are involved in generating diagrams or formatting the final PDF for the pre-print production process. In short, there is only one or two people who will need to have access to the following set of tools, and, in a normal publishing house, all of these functions would likely be performed by a separate "Production" team.

Omnigraffle

Many of the diagrams in this book have been generated using a OSX-specific tool named Omnigraffle.

If you are interested in helping us create diagrams don't feel compelled to purchase a copy of Omnigraffle. Send us a rough outline of your diagram, and Sonatype will gladly transform your idea into a diagram if your contributions are accepted into the book.

Adobe Photoshop

All of the screen shots are generated using simple screen capture tools. The resulting raw images (PNGs) are then processed using a set of very simple Photoshop macros. These macros add a border to each screen shot and apply a standard drop shadow. Once the drop shadow has been applied, these macros then save a 72 dpi PNG image for the HTML version of the book in addition to a 150 dpi PDF image for the printed version of the book,

While Adobe Photoshop is a capable (and somewhat formidable) graphics manipulation tool, Sonatype is exploring alternatives to using this commercial utility in the content generation process. Alternatives currently being investigated are open source packages such as GIMP or systems which can rely on ImageMagick for scripted conversion of raw screen shots to multiple web and print image formats.

If you are interested in contributing, but you do not want to bother with the process of formatting images for both web and print, Sonatype welcomes contributions which include raw screen captures. We can take care of the formatting.

Adobe Illustrator

The book cover, the promotional material insert, and the print binder for Lulu are all generated with Adobe Illustrator. Adobe Illustrator can open and edit PDFs natively, and can be used to generate the static PDFs which are concatenated together to produce the final book output.

D.4 How to Build the Book

You know what, this book doesn't really have a "build" as much as it has a collection of light-weight bash scripts which are used to invoke Asciidoc and a2x. If you are interested in building the book, we've had good success on Mac OSX and Ubuntu. Any other operation system and you are on your own.

- Clone the book's Git repository. To clone this book's repository execute the following command at a command-line:

```
$ git clone git@github.com:sonatype/nexus-book.git  
...a bunch of Git output...
```

Running this command will create a sub-directory named `nexus-book` which is a copy of this book's source.

- `./build.sh`

This will create a single-page HTML version, a chunked HTML version, and a PDF version of the book.

D.5 Subscribing to the Book Developers List

Sonatype maintains a Book Developers mailing list as a single mailing list for contributors, authors, and editors working on any of our Sonatype Open Books. This is a high volume list which contains both discussion and automated emails from GitHub and our Sonatype Matrix continuous integration server.

To subscribe to this mailing list, send and email to: book-dev-subscribe@sonatype.org

Appendix E

Copyright

Copyright © 2011 Sonatype, Inc. All rights reserved.

Online version published by Sonatype, Inc,

Nexus™, Nexus Professional™, and all Nexus-related logos are trademarks or registered trademarks of Sonatype, Inc., in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle, Inc., in the United States and other countries.

IBM® and WebSphere® are trademarks or registered trademarks of International Business Machines, Inc., in the United States and other countries.

Eclipse™ is a trademark of the Eclipse Foundation, Inc., in the United States and other countries.

Apache and the Apache feather logo are trademarks of The Apache Software Foundation.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Sonatype, Inc. was aware of a trademark

claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Appendix F

Creative Commons License

This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States license. For more information about this license, see <http://creativecommons.org/licenses/by-nc-nd/3.0/us/>. You are free to share, copy, distribute, display, and perform the work under the following conditions:

- You must attribute the work to Sonatype, Inc. with a link to <http://www.sonatype.com>
- You may not use this work for commercial purposes.
- You may not alter, transform, or build upon this work.

If you redistribute this work on a web page, you must include the following link with the URL in the about attribute listed on a single line (remove the backslashes and join all URL parameters):

```
<div xmlns:cc="http://creativecommons.org/ns#"  
about="http://creativecommons.org/license/results-one?q_1=2&q_1=1\"  
&field_commercial=n&field_derivatives=n&field_jurisdiction=us\"  
&field_format=StillImage&field_worktitle=Repository%3A+\Management\"  
&field_attribute_to_name=Sonatype%2C+Inc.\\"  
&field_attribute_to_url=http%3A%2F%2Fwww.sonatype.com\"  
&field_sourceurl=http%3A%2F%2Fwww.sonatype.com%2Fbook\"  
&lang=en_US&language=en_US&n_questions=3">  
<a rel="cc:attributionURL" property="cc:attributionName"  
href="http://www.sonatype.com">Sonatype, Inc.</a> /  
<a rel="license"
```

```
href="http://creativecommons.org/licenses/by-nc-nd/3.0/us/"> >  
CC BY-NC-ND 3.0</a>  
</div>
```

When downloaded or distributed in a jurisdiction other than the United States of America, this work shall be covered by the appropriate ported version of Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 license for the specific jurisdiction. If the Creative Commons Attribution-Noncommercial-No Derivative Works version 3.0 license is not available for a specific jurisdiction, this work shall be covered under the Creative Commons Attribution-Noncommercial-No Derivative Works version 2.5 license for the jurisdiction in which the work was downloaded or distributed. A comprehensive list of jurisdictions for which a Creative Commons license is available can be found on the Creative Commons International web site at <http://creativecommons.org/international>

If no ported version of the Creative Commons license exists for a particular jurisdiction, this work shall be covered by the generic, unported Creative Commons Attribution-Noncommercial-No Derivative Works version 3.0 license available from <http://creativecommons.org/licenses/by-nc-nd/3.0/>

F.1 Creative Commons BY-NC-ND 3.0 US License

Creative Commons Attribution-NonCommercial-NoDerivs 3.0 United States

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

- a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with one or more other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

- b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
 - c. "Licensor" means the individual, individuals, entity or entities that offers the Work under the terms of this License.
 - d. "Original Author" means the individual, individuals, entity or entities who created the Work.
 - e. "Work" means the copyrightable work of authorship offered under the terms of this License.
 - f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.
2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.
 3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:
 - a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works; and,
 - b. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(d) and 4(e).

1. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:
 - a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource

Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of a recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. When You distribute, publicly display, publicly perform, or publicly digitally perform the Work, You may not impose any technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any credit as required by Section 4(c), as requested.

- b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.
- c. If You distribute, publicly display, publicly perform, or publicly digitally perform the Work (as defined in Section 1 above) or Collective Works (as defined in Section 1 above), You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear, if a credit for all contributing authors of the Collective Work appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this clause for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

2. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR

OFFERS THE WORK AS-IS AND ONLY TO THE EXTENT OF ANY RIGHTS HELD IN THE LICENSED WORK BY THE LICENSOR. THE LICENSOR MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MARKETABILITY, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

1. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
2. Termination
 - a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works (as defined in Section 1 above) from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
 - b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.
3. Miscellaneous
 - a. Each time You distribute or publicly digitally perform the Work (as defined in Section 1 above) or a Collective Work (as defined in Section 1 above), the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
 - b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
 - c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

- d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

F.2 Creative Commons Notice

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, Creative Commons does not authorize the use by either party of the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time. For the avoidance of doubt, this trademark restriction does not form part of this License.

Creative Commons may be contacted at <http://creativecommons.org/>.

/* Local Variables: // ispell-personal-dictionary: "ispell.dict" // End: */
