REST and Integration API

Table of Contents

Available in Nexus Repository OSS and Nexus Repository Pro

REST API

Nexus Repository Manager leverages a simple UI interface to generate the comprehensive REST documentation from the available endpoints. The REST API can be used to integrate the repository manager with external systems. The interface is fully interactive, where parameters can be filled out and REST calls made directly through the UI to see the results in the browser.

As of NXRM 3.6.1, this interface is available under the *API* item via the *System* sub menu of the *Administration* menu.

For older versions of NXRM (3.3.0 through 3.6.0), use the following:

```
<nexus_url>/swagger-ui/
```

A comprehensive listing of REST API endpoints and functionality is documented through the UI. The following highlights noteworthy functionality:

- Search - component search functionality
- Assets - direct access the assets/binaries and associated metadata in a repository
- Components - direct access to language native logical grouping of files in a repository, e.g. maven2 groupId, artifactId, version
- Script - for provisioning and other complex tasks
- Tasks - available administrative tasks for the repository manager

Topics in this section:

Scripting API

Disabled Groovy Scripting By Default

NEW IN RELEASE 3.21.2

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

For more information (including how to re-enable Groovy scripting), see NEXUS-23205[1].

Nexus Repository Manager also provides a scripting API. Scripts can be written to perform custom tasks that can't be handled directly through the UI or the REST API. NXRM scripts are written in the Groovy programming language. More information about Groovy can be found at http://groovy-lang.org/.

There are several methods to run scripts, including:

- To run a script from the Nexus Repository UI, you must create a scheduled task. Navigate to *Tasks* from the *Administration* menu, and create a task of type *Admin - Execute script*. Enter the script source into the *Source* form field. Set the task frequency to *Manual* (or another setting to execute on a schedule) and save. The script can then be executed by selecting it from the list of tasks and pressing the *Run* button.
- A script may be uploaded and run through the REST Script API(see page 39)

Nexus provides the following APIs to scripts:

- Core API: allows a script to set global configuration options
- Security API: allows a script to add users, roles, and privileges
- Blob Store API: allows a script to create new blob stores
- Repository API: provides a simple, format-specific interface for creating repositories

[1] https://issues.sonatype.org/browse/NEXUS-23205

Sample scripts and javadoc API documentation can be found at https://github.com/sonatype-nexus-community/nexus-scripting-examples.

For recommendations on configuring a stable scripting environment and tips from Sonatype, see this knowledge base article[2].

Assets API

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

Introduction

This set of endpoints is for interacting with assets directly.

Endpoints

List Assets

```
GET /service/rest/v1/assets
```

This endpoint allows us to iterate through a listing of assets contained in a given repository.

Let's get a listing of the assets in the `maven-central` repository:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/assets?repository=maven-central'
```

This produces a response that is the first pages of many assets in that repository:

[2] https://support.sonatype.com/hc/en-us/articles/115015812727-Nexus-3-Groovy-Script-development-environment-setup

Example Response

```
{
  "items" : [ {
    "downloadUrl" : "http://localhost:8081/repository/maven-central/asm/asm/3.1/asm-3.1-sources.jar",
    "path" : "asm/asm/3.1/asm-3.1-sources.jar",
    "id" : "bWF2ZW4tY2VudHJhbDozZjVjYWUwMTc2MDIzM2I2MTRkZWJlOTExOGFjNjMwOQ",
    "repository" : "maven-central",
    "format" : "maven2",
    "checksum" : {
      "sha1" : "2eaa4de56203f433f287a6df5885ef9ad3c5bcae",
      "md5" : "0c0e3e83d9ce7907334c5eb519dab10c"
    }
  },
...
  ],
  "continuationToken" : "3f5cae01760233b6506547dc7be10e0b"
}
```

This endpoint uses a pagination[3] strategy that can be used to iterate through all the assets if desired.

Note that the ordering of assets is consistent across multiple queries and is not alphabetical.

Get Asset

```
GET /service/rest/v1/assets/{id}
```

This endpoint allows us to get the details of an individual asset.

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/assets/
bWF2ZW4tY2VudHJhbDozZjVjYWUwMTc2MDIzM2I2MjRiOTEwMmMwMmNiYmU4YQ'
```

[3] https://help.sonatype.com/display/NXRM3M/Pagination

Example Response

```
{
  "downloadUrl" : "http://localhost:8081/repository/maven-central/org/sonatype/nexus/buildsupport/nexus-buildsupport-metrics/2.9.1-02/nexus-buildsupport-metrics-2.9.1-02.pom",
  "path" : "org/sonatype/nexus/buildsupport/nexus-buildsupport-metrics/2.9.1-02/nexus-buildsupport-metrics-2.9.1-02.pom",
  "id" : "bWF2ZW4tY2VudHJhbDozZjVjYWUwMTc2MDIzM2I2MjRiOTEwMmMwMmNiYmU4YQ",
  "repository" : "maven-central",
  "format" : "maven2",
  "checksum" : {
    "sha1" : "a3bf672b3ea844575acba3b84790e76ed86a7c66",
    "md5" : "49e439c814c3098450dc4bbee952463f"
  }
}
```

Delete Asset

```
DELETE /service/rest/v1/assets/{id}
```

This endpoint can be used to delete an individual asset.

```
curl -u admin:admin123 -X DELETE 'http://localhost:8081/service/rest/v1/assets/bWF2ZW4tY2VudHJhbDozZjVjYWUwMTc2MDIzM2I2MjRiOTEwMmMwMmNiYmU4YQ'
```

A successful deletion will return no content, for example:

Example Response

```
HTTP/1.1 204 No Content
Date: Fri, 19 Jan 2018 20:41:47 GMT
...
```

Blob Store API

Introduction

The Blob Store[4] exposes HTTP endpoints for creating, updating, deleting, retrieving and listing blob stores.

[4] https://help.sonatype.com/display/NXRM3/Storage+Guide

Please see the *API* section (found in the *Administration* section under *System* → *API*) of your Nexus Repository Manager instance for detailed information about the *Blob Store* API including sample requests and responses.

Endpoints

NEW IN 3.19 The blobstores REST API endpoints can be used to accomplish the following:

- List blob stores
- Delete blob store
- Create blob store
- Get blob store configuration
- Update blob store
- Get blob store quota[5]

Components API

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

Introduction

This set of endpoints is for interacting with components directly.

Endpoints

List Components

```
GET /service/rest/v1/components
```

This endpoint allows us to iterate through a listing of components contained in a given repository.

Let's get a listing of the components in the

`maven-central`

repository:

[5] https://help.sonatype.com/display/NXRM3/Repository+Management#RepositoryManagement-AddingaSoftQuota

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/components?repository=maven-central'
```

Typically this will produce a response that is the first page of many components in that repository, as such:

Example Response

```
{
  "items" : [ {
    "id" : "bWF2ZW4tY2VudHJhbDo4ODQ5MWNkMWQxODVkZDEzNjMyNjhmMjIwZTQ1ZDdkZQ",
    "repository" : "maven-central",
    "format" : "maven2",
    "group" : "com.google.guava",
    "name" : "guava",
    "version" : "21.0",
    "assets" : [ {
      "downloadUrl" : "http://localhost:8081/repository/maven-central/com/google/guava/guava/21.0/
guava-21.0.jar",
      "path" : "com/google/guava/guava/21.0/guava-21.0.jar",
      "id" : "bWF2ZW4tY2VudHJhbDozZjVjYWUwMTc2MDIzM2I2MzA4OThiZjZmZTFkOTE2NA",
      "repository" : "maven-central",
      "format" : "maven2",
      "checksum" : {
        "sha1" : "3a3d111be1be1b745edfa7d91678a12d7ed38709",
        "md5" : "ddc91fd850fa6177c91aab5d4e4d1fa6"
      }
    }, {
      "downloadUrl" : "http://localhost:8081/repository/maven-central/com/google/guava/guava/21.0/
guava-21.0.jar.sha1",
      "path" : "com/google/guava/guava/21.0/guava-21.0.jar.sha1",
      "id" : "bWF2ZW4tY2VudHJhbDpmODk4YjM5MDNjYjk5YzU5MDc0MDFlYzRjNjVlNjU5OQ",
      "repository" : "maven-central",
      "format" : "maven2",
      "checksum" : {
        "sha1" : "a1ff60cb911e1f64801c03d03702044d10c9bdd3",
        "md5" : "e34b8695ede1677ba262411d757ea980"
      }
    }, {
      "downloadUrl" : "http://localhost:8081/repository/maven-central/com/google/guava/guava/21.0/
guava-21.0.pom",
      "path" : "com/google/guava/guava/21.0/guava-21.0.pom",
      "id" : "bWF2ZW4tY2VudHJhbDpkMDY0ODA0YThlZDVhZDZlOWJjNDgzOGE1MzM2OGZlZg",
      "repository" : "maven-central",
      "format" : "maven2",
      "checksum" : {
        "sha1" : "fe4fa08a8c0897f9896c7e278fb397ede4a2feed",
        "md5" : "5c10f97af2ce9db54fa6c2ea6997a8d7"
      }
    }, {
      "downloadUrl" : "http://localhost:8081/repository/maven-central/com/google/guava/guava/21.0/
guava-21.0.pom.sha1",
      "path" : "com/google/guava/guava/21.0/guava-21.0.pom.sha1",
      "id" : "bWF2ZW4tY2VudHJhbDplMDE4OGVkMDcyOGZhNjhmZDA3NDdkNjlhZDNmZjI5Nw",
      "repository" : "maven-central",
      "format" : "maven2",
      "checksum" : {
        "sha1" : "992b43ab7b3a061be47767e910cab58180325abc",
        "md5" : "33aed29aa0bb4e03ea7854066a5b4738"
      }
    } ]
  },
...
  ],
  "continuationToken" : "88491cd1d185dd136f143f20c4e7d50c"
}
```

This endpoint uses a pagination(see page 55) strategy that can be used to iterate through all the components if desired.

Note that the ordering of the components is consistent across multiple queries and it is not alphabetical.

Get Component

```
GET /service/rest/v1/components/{id}
```

This endpoint allows us to get details about an individual component.

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/components/
bWF2ZW4tY2VudHJhbDo4ODQ5MWNkMWQxODVkZDEzNjYwYmUwMjE1MjI2NGUwZQ'
```

Example Response

```
{
  "id" : "bWF2ZW4tY2VudHJhbDo4ODQ5MWNkMWQxODVkZDEzNjYwYmUwMjE1MjI2NGUwZQ",
  "repository" : "maven-central",
  "format" : "maven2",
  "group" : "org.apache.httpcomponents",
  "name" : "httpcomponents-client",
  "version" : "4.3.5",
  "assets" : [ {
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/apache/httpcomponents/
httpcomponents-client/4.3.5/httpcomponents-client-4.3.5.pom",
    "path" : "org/apache/httpcomponents/httpcomponents-client/4.3.5/httpcomponents-client-4.3.5.pom",
    "id" : "bWF2ZW4tY2VudHJhbDozZjVjYWUwMTc2MDIzM2I2YTFhOGUxOGQxZmFkOGM3Mw",
    "repository" : "maven-central",
    "format" : "maven2",
    "checksum" : {
      "sha1" : "95d80a44673358a5dcbcc2f510770b9f93fe5eba",
      "md5" : "f4769c4e60799ede664414c26c6c5c9d"
    }
  }, {
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/apache/httpcomponents/
httpcomponents-client/4.3.5/httpcomponents-client-4.3.5.pom.sha1",
    "path" : "org/apache/httpcomponents/httpcomponents-client/4.3.5/httpcomponents-client-4.3.5.pom.sha1",
    "id" : "bWF2ZW4tY2VudHJhbDpmODk4YjM5MDNjYjk5YzU5ZDU3YjFlYjE0MzM1ZTcwMQ",
    "repository" : "maven-central",
    "format" : "maven2",
    "checksum" : {
      "sha1" : "6b98f5cef5d7102f8f45215bdcf48dc843d060af",
      "md5" : "f3b3ac640853fcb887621d13029a1747"
    }
  } ]
}
```

Upload Component

```
POST /service/rest/v1/components
```

This endpoint allows us to upload a component to a specified repository. Some formats allow to include multiple assets in a single component (please refer to Uploading Components[6] and more information below).

```
curl -v -u admin:admin123 -X POST 'http://localhost:8081/service/rest/v1/components?repository=maven-
releases' -F maven2.groupId=com.google.guava -F maven2.artifactId=guava -F maven2.version=24.0-jre -F
maven2.asset1=@guava-24.0-jre.jar -F maven2.asset1.extension=jar -F maven2.asset2=@guava-24.0-jre-
sources.jar -F maven2.asset2.classifier=sources -F maven2.asset2.extension=jar
```

[6] https://help.sonatype.com/display/NXRM3/Uploading+Components

A successful upload will return no content, as follows:

```
HTTP/1.1 204 No Content
Date: Fri, 19 Jan 2018 20:26:13 GMT
...
```

The parameters expected by the endpoint depend on the format of the repository where we are going to upload our component.

Maven

Maven format allows multiple assets to be uploaded as part of a single component. To upload multiple assets just follow the information from a table describing the given format and replace

`assetN`

with multiple instances of it (e.g. `asset1`, `asset2`, etc.):

Field name	Field type	Required?	Description
maven2.groupId	String	Yes, unless a POM asset is included in the upload	Group ID of the component
maven2.artifactId	String	Yes, unless a POM asset is included in the upload	Artifact ID of the component
maven2.version	String	Yes, unless a POM asset is included in the upload	Version of the component
maven2.generate-pom	Boolean	No	Whether the Nexus Repository Manager should generate a POM file based on above component coordinates provided
maven2.packaging	String	No	Define component packaging (e.g. jar, ear)
maven2.assetN	File	Yes, at least one	Binary asset
maven2.assetN.extension	String	Yes	Extension of the corresponding `assetN` asset
maven2.assetN.classifier	String	No	Classifier of the corresponding `assetN` asset

Examples

Uploading a jar and Automatically Creating a pom File

For this example we have a single jar file from a third-party vendor and no project POM file from that third party. We have decided our own Maven 2 coordinates and will ask Nexus to automatically generate and store a POM file for us.

Example Maven jar Component Upload With Automatically Created pom File Using curl

```
curl -v -u admin:admin123 -F "maven2.generate-pom=true" -F "maven2.groupId=com.example" -F
"maven2.artifactId=commercial-product" -F "maven2.packaging=jar" -F "version=1.0.0" -F "maven2.asset1=@/
absolute/path/to/the/local/file/product.jar;type=application/java-archive" -F "maven2.asset1.extension=jar"
 "http://localhost:8081/service/rest/v1/components?repository=maven-third-party"
```

- **maven2.generate-pom=true** tells Nexus to generate a minimal POM file for us based on the groupId, artifactId, version and packaging we provide
- replace "admin:admin123" with your own valid username:password
- replace "/absolute/path/to/the/local/file/product.jar " with the local path to the file you are uploading, however leave the leading @ so that curl understands how to find this file
- *;type=application/java-archive* is optional
- the repository **maven-third-party** must already exist or there will be an error
- -v makes the command verbose to help you verify a successful response is returned. Otherwise you wouldn't be able to easily tell if it worked or returned an HTTP error code

Upload a POM and associated JAR File

For this example we have a single jar file and we have already created a minimal project pom.xml which includes groupId, artifactId, version elements defined inside of it. Nexus will acccept both the POM and jar file in a single upload operation, and will parse the Maven 2 coordinates from the uploaded POM file and install the artifacts to the correct locations inside the maven-releases respository. Notice how the command need not specify the Maven coordinates.

Example pom and jar Component Upload Using curl

```
curl -v -u admin:admin123 -F "maven2.generate-pom=false" -F "maven2.asset1=@/absolute/path/to/the/local/
file/pom.xml" -F "maven2.asset1.extension=pom" -F "maven2.asset2=@/absolute/path/to/the/local/file/
product-1.0.0.jar;type=application/java-archive" -F "maven2.asset2.extension=jar" "http://localhost:8081/
service/rest/v1/components?repository=maven-releases"
```

Raw

Raw supports uploading multiple assets within a request

Field name	Field type	Required?	Description
raw.directory	String	Yes	Destination for upload files (e.g. /path/to/files)
raw.assetN	File	Yes, at least one	Binary asset
raw.assetN.filename	String	Yes	Filename to be used for the corresponding `assetN` asset

PyPI

Field name	Field type	Required?	Description
pypi.asset	File	Yes	Binary asset

RubyGems

Field name	Field type	Required?	Description
rubygems.asset	File	Yes	Binary asset

NuGet

Field name	Field type	Required?	Description
nuget.asset	File	Yes	Binary asset

NPM

Field name	Field type	Required?	Description
npm.asset	File	Yes	Binary asset

Delete Component

```
DELETE /service/rest/v1/components/{id}
```

This endpoint can be used to delete an individual component along with the associated assets.

```
curl -u admin:admin123 -X DELETE 'http://localhost:8081/service/rest/v1/components/
bWF2ZW4tY2VudHJhbDo4ODQ5MWNkMWQxODVkZDEzNjYwYmUwMjE1MjI2NGUwZQ'
```

A successful deletion will return no content, as follows:

Example Response

```
HTTP/1.1 204 No Content
Date: Fri, 19 Jan 2018 20:26:13 GMT
...
```

Email API

Introduction

The Email API exposes a set of endpoints for specifying and viewing SMTP server settings used by NXRM for sending email[7].

Please see the *API* section (found in the *Administration* section under *System → API*) of your Nexus Repository Manager instance for detailed information about the *email* API including sample requests and responses.

Endpoints

NEW IN 3.19 The email REST API endpoints can be used to accomplish the following:

- Get email configuration
- Set email configuration
- Disable email configuration
- Verify email configuration

IQ Server API

[7] https://help.sonatype.com/display/NXRM3/System+Configuration#SystemConfiguration-Email/SMTPServerConfiguration

Introduction

The IQ Server API exposes HTTP endpoints for creating, updating, deleting and viewing of IQ server connection details.

Please see the *API* section (found in the *Administration* section under *System* → *API*) of your Nexus Repository Manager instance for detailed information about the IQ API including sample requests and responses.

Endpoints

NEW IN 3.19 The IQ[8] REST API endpoints can be used to accomplish the following:

- Get IQ server configuration
- Update IQ server configuration
- Disable IQ server configuration
- Enable IQ server configuration
- Verify IQ server connection

Licensing API

Introduction

The *Product Licensing* API exposes a set of endpoints for License Management[9].

Please see the *API* section (found in the *Administration* section under *System* → *API*) of your Nexus Repository Manager instance for detailed information about the *Product Licensing* API including sample requests and responses.

Endpoints

NEW IN 3.19 The license REST API endpoints can be used to accomplish the following:

- Get license
- Upload new license
- Delete license

[8] https://help.sonatype.com/display/NXIQ

[9] https://help.sonatype.com/display/NXRM3/License+Management

Lifecycle API

These operations are intended to be used with the guidance of Sonatype support. Usage without supervision of Sonatype is not supported or recommended.

Introduction

The Lifecycle API allow Nexus Repository Manager administrators to change the phase in which a NXRM instance is running. A lifecycle phase is a step in the start up process used to group similar components and ensure that their dependencies are started before them. In order of execution during startup, the phases are:

1. OFF - NXRM is completely off and the process is shut down.
2. KERNEL - The most basic parts of the node are running.
3. STORAGE - The databases and caches are setup.
4. RESTORE - Any restoring from backups happens in this phase.
5. UPGRADE - If any upgrades are needed, they will occur is this phase.
6. SCHEMAS - This creates any missing schemas in the database.
7. EVENTS - The event bus is initialized.
8. SECURITY - The parts of NXRM responsible for security are enabled.
9. SERVICES - This phase starts the all of the services, such as repositories.
10. CAPABILITIES - The capabilities are started; see Accessing and Configuring Capabilities[10]
11. TASKS - The task scheduler is enabled, and tasks may run.

Endpoints

Lifecycle Phase

```
GET /service/rest/v1/lifecycle/phase
```

This returns the current phase in which NXRM is running.

[10] https://help.sonatype.com/display/NXRM3M/System+Configuration#SystemConfiguration-AccessingandConfiguringCapabilities

```
curl -u admin:admin123 -X GET "http://localhost:8081/service/rest/v1/lifecycle/phase" -H "accept: text/
plain"
```

Example Response

```
TASKS
```

```
PUT /service/rest/v1/lifecycle/phase
```

This sets NXRM to run in a particular phase. Any features that are enabled by later phases are shut down. e.g. Setting NXRM to the CAPABILITIES phase will stop any tasks from running, since they only occur in the TASKS phase.

```
curl -u admin:admin123 -X PUT "http://localhost:8081/service/rest/v1/lifecycle/phase" -H "accept:
application/json" -H "Content-Type: text/plain" -d "SERVICES"
```

Lifecycle Bounce

```
PUT /service/rest/v1/lifecycle/bounce
```

This operation takes the instance down to the supplied Lifecycle phase, and then back to the phase in which it was originally running. This is used primarily to restart phase in one operation.

```
curl -u admin:admin123 -X PUT "http://localhost:8081/service/rest/v1/lifecycle/bounce" -H "accept:
application/json" -H "Content-Type: text/plain" -d "TASKS"
```

Maintenance API

Available in Nexus Repository Pro

Introduction

These REST endpoints allow the user to inspect and manipulate the state of a node in Orient database cluster. The expectation is that these will be used mainly for troubleshooting and recovery operations.

Possible Database Names: accesslog, component, config, security

Endpoints

Check Database

```
PUT /service/rest/v1/maintenance/{databaseName}/check
```

Checks database pages for corruption and checks that indices cover all records (i.e. no duplicates).

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/check
```

Example Response

```
{
  "pageCorruption" : false,
  "indexErrors" : 0
}
```

Reinstall Database

> Only available in HA-C

```
PUT /service/rest/v1/maintenance/{databaseName}/reinstall
```

Attempts to reinstall the full database from the rest of the cluster.

> ⚠ The user can't force a reinstall from a specific node, but typically Orient will choose the node that most recently took a backup or the oldest member (assuming that's not the node requesting the reinstall)

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/reinstall
```

Example Response

```
{
  "reinstalled" : true
}
```

Repair Database

```
PUT /service/rest/v1/maintenance/{databaseName}/repair
```

Attempts to repair any corrupt database pages and rebuilds indices.

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/repair
```

Example Response

```
{
  "indexErrorsRemaining" : 0,
  "repairedPages" : 0
}
```

Database Role

ⓘ Only available in HA-C

This attribute represents the role of the database node in an Orient cluster. The role of MASTER represents a fully writable database node in an Orient cluster which participates in the `writeQuorum`. The role of REPLICA represents a database node in read-only mode which is accepting only idempotent commands, e.g. read and queries.

Possible Roles: MASTER, REPLICA

Get Role

```
GET /service/rest/v1/maintenance/{databaseName}/role
```

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/maintenance/component/role
```

Example Response

```
MASTER
```

Set Role

```
PUT /service/rest/v1/maintenance/{databaseName}/role
```

When setting a database node's role to REPLICA an override is added to Orient's configuration so that node will always be a REPLICA, regardless of the cluster's default role. When the database node is set back to MASTER that override is removed, so it will defer to the cluster's default role - which is MASTER unless it's been frozen.

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/role -H "Content-Type: text/plain" -d "MASTER"
```

Example Response

Database Status

ⓘ Only available in HA-C

This attribute represents the status of the database node in an Orient cluster. The status of ONLINE represents a database node in a normal state which is fully participating in the cluster. The status of OFFLINE represents a database node that is not currently participating in the cluster. The status of NOT_AVAILABLE represents a database node that is not currently available for some reason.

Possible Statuses: ONLINE, OFFLINE, NOT_AVAILABLE

Get Status

```
GET /service/rest/v1/maintenance/{databaseName}/status
```

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/maintenance/component/status
```

Example Response

```
ONLINE
```

Set Status

```
PUT /service/rest/v1/maintenance/{databaseName}/status
```

⚠ Setting the database status to NOT_AVAILABLE triggers an automatic delta-sync with the rest of the cluster. If this is successful, it will move back to ONLINE.

```
curl -u admin:admin123 -X PUT http://localhost:8081/service/rest/v1/maintenance/component/status -H
"Content-Type: text/plain" -d "ONLINE"
```

Example Response

Nodes API

ⓘ **Available in Nexus Repository Pro**

Introduction

These endpoints allow us to get information about the nodes in the current cluster and update the friendly names of individual nodes. The list of nodes returned is a view of the state of the system and cannot be edited through this API. From this API, nodes can be assigned friendly names, but that is currently the only editable attribute. The friendly names may provide a more convenient mechanism to identify individual nodes for administrative activities.

Endpoints

Get Nodes

```
GET /service/rest/v1/nodes
```

This endpoint gives us a listing of the nodes currently in the cluster.

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/nodes
```

The information returned includes a unique `nodeIdentity` for each node along with the current socket address binding and the current friendly name of the node:

```
[ {
  "nodeIdentity" : "20550283-C2239399-2DC0307C-DE78EE76-0670E0BC",
  "socketAddress" : "/172.18.0.2:5701",
  "friendlyName" : null
}, {
  "nodeIdentity" : "2840754A-E8053498-0DA01A51-9A7582DD-A52D5279",
  "socketAddress" : "/172.18.0.3:5701",
  "friendlyName" : null
}, {
  "nodeIdentity" : "A54F51B3-F0EBD6F7-23D4C3A0-ABA922CD-9DCE73EA",
  "socketAddress" : "/172.18.0.4:5701",
  "friendlyName" : null
} ]
```

Update Nodes

```
PUT /service/rest/v1/nodes
```

This endpoint allows us to update friendly names for nodes in the cluster:

```
curl -u admin:admin123 -X PUT --header 'Content-Type: application/json' \
 http://localhost:8081/service/rest/v1/nodes \
 -d '{"nodeIdentity" : "20550283-C2239399-2DC0307C-DE78EE76-0670E0BC","socketAddress" : "/
172.18.0.2:5701","friendlyName" : "node1"}'
```

If successful, this operation returns the updated information for the node:

```
{
  "nodeIdentity" : "20550283-C2239399-2DC0307C-DE78EE76-0670E0BC",
  "socketAddress" : "/172.18.0.2:5701",
  "friendlyName" : "node1"
}
```

Read-Only API

Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

This set of endpoints is used to enable and disable read-only mode on the underlying database. This is useful when conducting some maintenance activities.

Endpoints

Get State

```
GET /service/rest/v1/read-only
```

This endpoint allows us to query the current read-only state of the database.

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/read-only
```

The resulting JSON document contains information about whether or not the database is currently "frozen" (i.e. in read-only mode), the reason why it may have been frozen, and whether or not the freeze was system initiated:

Example Response

```
{
  "frozen" : false,
  "summaryReason" : "",
  "systemInitiated" : false
}
```

Enable

```
POST /service/rest/v1/read-only/freeze
```

This endpoint allows us to put the database into read-only mode.

```
curl -u admin:admin123 -X POST http://localhost:8081/service/rest/v1/read-only/freeze
```

After issuing the freeze action, the database should be in read-only mode.

```
$ curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/read-only
{
  "frozen" : true,
  "summaryReason" : "activated by an administrator at 2018-01-22 15:04:05 +00:00",
  "systemInitiated" : false
}
```

Release

```
POST /service/rest/v1/read-only/release
```

This endpoint allows us to take the database out of read-only mode.

```
curl -u admin:admin123 -X POST http://localhost:8081/service/rest/v1/read-only/release
```

After issuing the release action, the database should no longer be in read-only mode.

```
$ curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/read-only
{
  "frozen" : false,
  "summaryReason" : "",
  "systemInitiated" : false
}
```

Force Release

```
POST /service/rest/v1/read-only/force-release
```

In certain rare instances we may need to forcibly take the database out of read-only mode. This endpoint allows us to do that.

```
curl -u admin:admin123 -X POST http://localhost:8081/service/rest/v1/read-only/force-release
```

After issuing the force-release action, the database should no longer be in read-only mode.

```
$ curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/read-only
{
  "frozen" : false,
  "summaryReason" : "",
  "systemInitiated" : false
}
```

Repositories API

Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

This set of endpoints is for interacting with repositories directly.

Endpoints

List Repositories

```
GET /service/rest/v1/repositories
```

This endpoint allows us to iterate through a listing of repositories a user has browse access to.

Let's get a listing of the repositories that the user `admin` can browse:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/repositories'
```

This produces a response that is a listing of all repositories the user is allowed to browse:

Example Response

```
[
  {
    "name": "nuget.org-proxy",
    "format": "nuget",
    "type": "proxy",
    "url": "http://localhost:8081/repository/nuget.org-proxy",
    "attributes" : {
      "proxy" : {
        "remoteUrl" : "https://www.nuget.org/api/v2/"
      }
    }
  },
  {
    "name": "maven-releases",
    "format": "maven2",
    "type": "hosted",
    "url": "http://localhost:8081/repository/maven-releases"
  },
...
]
```

This endpoint returns all repositories and does not allow for pagination.

Note that the ordering of repositories is consistent across multiple queries and is not alphabetical.

Search API

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

Introduction

The Search API facilitates searching for components and assets in addition to downloading a specific asset.

Endpoints

Search Components

```
GET /service/rest/v1/search
```

This endpoint is useful for finding components based on general criteria (e.g. group, name, version) as well as format specific attributes (e.g. maven.baseVersion, npm.scope, yum.architecture). The search uses the same mechanism used by the Repository Manager UI in order to find components. Details about the criteria that can be used for searching are described in the generated documentation in Repository Manager under the administrative view System/API (`<nexus_url>/#admin/system/api`).

The search takes the form of a GET request against the endpoint which returns a JSON document with information about the components (and associated assets) that were found.

As an example, we can search for all components in the maven-central repository which have a group of `org.osgi:`

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search?repository=maven-central&group=org.osgi'
```

This returns a JSON document containing a list of items that correspond to the components found during the search:

Example Response

```
{
  "items" : [ {
    "id" : "bWF2ZW4tY2VudHJhbDoyZTQ3ZGRhMGYxYjU1NWUwNzE1OWRjOWY5ZGQzZmVmNA",
    "repository" : "maven-central",
    "format" : "maven2",
    "group" : "org.osgi",
    "name" : "org.osgi.core",
    "version" : "4.3.1",
    "assets" : [ {
      "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/
org.osgi.core-4.3.1-sources.jar",
      "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1-sources.jar",
      "id" : "bWF2ZW4tY2VudHJhbDplMDE4OGVkMDcyOGZhNjhmNDExNzU2OGU1MjQ2NjZiYg",
      "repository" : "maven-central",
      "format" : "maven2",
      "checksum" : {
        "sha1" : "80bfafcf783988442b3a58318face1d2132db33d",
        "md5" : "87ee0258b79dc852626b91818316b9c3"
      }
    }, {
      "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/
org.osgi.core-4.3.1.jar",
      "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.jar",
      "id" : "bWF2ZW4tY2VudHJhbDpkMDY0ODA0YThlZDVhZDZlNjhmZGU5MWNmM2NiZTgzMw",
      "repository" : "maven-central",
      "format" : "maven2",
      "checksum" : {
        "sha1" : "5458ffe2ba049e76c29f2df2dc3ffccddf8b839e",
        "md5" : "8053bbc1b55d51f5abae005625209d08"
      }
    }, {
      "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/
org.osgi.core-4.3.1.pom",
      "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.pom",
      "id" : "bWF2ZW4tY2VudHJhbDo2NTRiYjdkMGE1OTIxMzg1OWZhMTVkMzNmYWU1ZmY3OA",
      "repository" : "maven-central",
      "format" : "maven2",
      "checksum" : {
        "sha1" : "79391fc69dd72ad1fd983d01b4572f93f644882b",
        "md5" : "3d87a59bcdb4b131d9a63e87e0ed924a"
      }
    } ]
  } ],
  "continuationToken" : null
}
```

In this case we've found one component which has three assets associated with it. Each component and asset has an `id` that can be used with the component and asset specific endpoints. Since only one item was found, there is no need for pagination so the `continuationToken` has a value of `null` to signify that this is the last page of items. The `downloadUrl` can be used to directly download the asset from the repository manager.

This endpoint uses a pagination[11] strategy that can be used to iterate through all the search results if desired.

Search Assets

```
GET /service/rest/v1/search/assets
```

This endpoint is focused on searching for assets. All of the same search criteria are available as in the component search above, but only assets will be returned.

Let's again search the maven-central repository for assets which have a group of `org.osgi`:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search/assets?repository=maven-
central&group=org.osgi'
```

This returns a JSON document containing a list of items corresponding to the assets found during the search:

[11] https://help.sonatype.com/display/NXRM3M/Pagination

Example Response

```
{
  "items" : [ {
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/
org.osgi.core-4.3.1-sources.jar",
    "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1-sources.jar",
    "id" : "bWF2ZW4tY2VudHJhbDplMDE4OGVkMDcyOGZhNjhmNDExNzU2OGU1MjQ2NjZiYg",
    "repository" : "maven-central",
    "format" : "maven2",
    "checksum" : {
      "sha1" : "80bfafcf783988442b3a58318face1d2132db33d",
      "md5" : "87ee0258b79dc852626b91818316b9c3"
    }
  }, {
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/
org.osgi.core-4.3.1.jar",
    "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.jar",
    "id" : "bWF2ZW4tY2VudHJhbDpkMDY0ODA0YThlZDVhZDZlNjhmZGU5MWNmM2NiZTgzMw",
    "repository" : "maven-central",
    "format" : "maven2",
    "checksum" : {
      "sha1" : "5458ffe2ba049e76c29f2df2dc3ffccddf8b839e",
      "md5" : "8053bbc1b55d51f5abae005625209d08"
    }
  }, {
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/
org.osgi.core-4.3.1.pom",
    "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.pom",
    "id" : "bWF2ZW4tY2VudHJhbDo2NTRiYjdkMGE1OTIxMzg1OWZhMTVkMzNmYWU1ZmY3OA",
    "repository" : "maven-central",
    "format" : "maven2",
    "checksum" : {
      "sha1" : "79391fc69dd72ad1fd983d01b4572f93f644882b",
      "md5" : "3d87a59bcdb4b131d9a63e87e0ed924a"
    }
  } ],
  "continuationToken" : null
}
```

This shows us that we found three assets and that there are no additional pages of results.

This endpoint uses a pagination[12] strategy that can be used to iterate through all the search results if desired.

Search and Download Asset

```
GET /service/rest/v1/search/assets/download
```

[12] https://help.sonatype.com/display/NXRM3M/Pagination

This endpoint is specifically designed to search for one asset and then redirect the request to the `downloadUrl` of that asset.

Let's say we want to download the asset corresponding to a jar whose group is `org.osgi`, name is `org.osgi.core`, and version is `4.3.1`.

To achieve this, it is necessary that the search returns a single asset. Continuing our example, we can use the asset search endpoint above to refine our search until it returns a single asset:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search/assets?
group=org.osgi&name=org.osgi.core&version=4.3.1&maven.extension=jar&maven.classifier'
```

Note that in the above example, the last parameter "maven.classsifier" has no value. This indicates that the matched asset should have no classifier.

The response returned by the above looks like this:

Example Response

```
{
  "items" : [ {
    "downloadUrl" : "http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/
org.osgi.core-4.3.1.jar",
    "path" : "org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1.jar",
    "id" : "bWF2ZW4tY2VudHJhbDpkMDY0ODA0YThlZDVhZDZlNjhmZGU5MWNmM2NiZTgzMw",
    "repository" : "maven-central",
    "format" : "maven2",
    "checksum" : {
      "sha1" : "5458ffe2ba049e76c29f2df2dc3ffccddf8b839e",
      "md5" : "8053bbc1b55d51f5abae005625209d08"
    }
  } ],
  "continuationToken" : null
}
```

Then we can append `/download` to produce the URL we can use to download the asset:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search/assets/download?
group=org.osgi&name=org.osgi.core&version=4.3.1&maven.extension=jar&maven.classifier'
```

If successful, this will give us a 302 response and redirect us to the repository manager download URL for the asset.

Example Response

```
HTTP/1.1 302 Found
Content-Length: 0
Date: Fri, 19 Jan 2018 17:34:21 GMT
Location: http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/
org.osgi.core-4.3.1.jar
...
```

Note that the search parameters in the example contain a parameter (`maven.classifier`) with no value:

```
...?group=org.osgi&name=org.osgi.core&version=4.3.1&maven.extension=jar&maven.classifier
```

Specifying a parameter with no value is an indicator to the search endpoint to only return assets which correspondingly have no value for that parameter. This technique can be used filter down search results to a single asset for some formats such as Maven where the component contains multiple assets that match all search criteria specified right up to the extension. For example, a Maven component can have the actual JAR file for a library, plus have a sources JAR. In these cases, just specifying `maven.extension=jar` is not specific enough to return a single asset but we can further narrow the search results by specifying that we want only the main JAR file asset not the sources JAR by including an empty `maven.classifier` parameter.

Alternatively, let's now assume that instead of wanting the main `org.osgi.core` JAR we desire to search for and download the sources JAR (e.g. `org.osgi.core-4.3.1-sources.jar`). That is, we want to search for and download the asset corresponding to a jar whose group is `org.osgi`, name is `org.osgi.core`, version is 4.3.1 and maven.classifier is `sources`.

We can achieve this by using the asset search endpoint and including the `maven.classifier` with a specified value of `sources`:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/search/assets/download?
group=org.osgi&name=org.osgi.core&version=4.3.1&maven.classifier=sources'
```

If successful, this will give us a 302 response and redirect us to the repository manager download URL for the asset.

Example Response

```
HTTP/1.1 302 Found
Content-Length: 0
Date: Fri, 19 Jan 2018 17:34:21 GMT
Location: http://localhost:8081/repository/maven-central/org/osgi/org.osgi.core/4.3.1/org.osgi.core-4.3.1-
sources.jar
...
```

Downloading the Latest Version of an Asset

NEW IN 3.16.0

You can download the latest version of an asset by using the "sort=version" parameter. For example, this will download the highest semantic version available for the Maven group ID and artifact ID specified:

```
curl -X GET "http://localhost:8081/service/rest/v1/search/assets/download?sort=version&repository=maven-
snapshots&maven.groupId=org.foo.bar&maven.artifactId=project&maven.extension=jar" -H "accept: application/
json"
```

To get the latest version of a Maven snapshot, also include the base version:

```
curl -X GET "http://localhost:8081/service/rest/v1/search/assets/download?sort=version&repository=maven-
snapshots&maven.groupId=org.foo.bar&maven.artifactId=project&maven.baseVersion=1.2.3-
SNAPSHOT&maven.extension=jar" -H "accept: application/json"
```

Security Management API

Please see the *API* documentation (found in the *Administration* section under *System*) of your Nexus Repository Manager instance for complete documentation about the specific endpoints available in your local server.

Content Selectors

NEW IN 3.19 The content selectors REST API endpoints can be used to create and manage content selectors[13]:

- list the content selectors
- create or modify content selectors

LDAP

NEW IN 3.19 The LDAP[14] endpoints can be used to accomplish the following:

- List all LDAP servers
- Create LDAP server
- Retrieve the details of a single LDAP server
- Update LDAP server

[13] https://help.sonatype.com/display/NXRM3/Repository+Management#RepositoryManagement-ContentSelectors
[14] https://help.sonatype.com/display/NXRM3/LDAP

- Delete LDAP server
- Change LDAP server ordering

Privileges

NEW IN 3.19 The privileges REST API endpoints can be used to create and manage privileges[15]:

- list the privileges
- create or modify privileges

Roles

NEW IN 3.19 The roles REST API endpoints can be used to create and manage roles[16] and their permissions within NXRM:

- list the roles from any configured user source (internal or external)
- create or modify local roles and external role mappings

Users

NEW IN 3.17 The users REST API endpoints can be used to create and manage users[17] and their permissions within NXRM:

- search for users available to NXRM whether they were defined locally or from a configured authentication source such as LDAP
- create or modify local users
- change the roles associated with an external user
- reset an individual user token[18]
- reset all user tokens in the system

User Sources

NEW IN 3.17 This endpoint provides a list of the available users sources in the Nexus Repository Manager, these are used in other REST endpoints to indicate the source of certain types of entities, e.g. a user from an LDAP server.

[15] https://help.sonatype.com/display/NXRM3/Privileges

[16] https://help.sonatype.com/display/NXRM3/Roles

[17] https://help.sonatype.com/display/NXRM3/Users

[18] https://help.sonatype.com/display/NXRM3/Security+Setup+with+User+Tokens

Script API

Available in Nexus Repository OSS and Nexus Repository Pro

Introduction

This is a powerful scripting API that provides methods to simplify provisioning and executing other complex tasks in the repository manager. These APIs can be invoked from scripts that are published to the repository manager and executed within the application server.

More detailed documentation is available in the following topics:

- Writing Scripts(see page 43)
- Managing and Running Scripts(see page 45)
- Examples(see page 47)

Endpoints

Add Script

```
POST /service/rest/v1/script
```

This endpoint allows us to create a utility script within the repository manager.

As an example, let's create a simple script that simply logs "Hello, World!":

```
curl -u admin:admin123 -X POST --header 'Content-Type: application/json' \
 http://localhost:8081/service/rest/v1/script \
 -d @helloWorld.json
```

The JSON document we send includes the unique name of the script, the contents of the script itself, and the language type of the script (in this case groovy).

helloWorld.json

```
{
  "name": "helloWorld",
  "content": "log.info('Hello, World!')",
  "type": "groovy"
}
```

If you are using Windows and want to use Powershell to add a script you can use the following commands:

```
$bytes = [System.Text.Encoding]::UTF8.GetBytes("admin:admin123")
$cred = [System.Convert]::ToBase64String($bytes)
$uri = "http://localhost:8081/service/rest/v1/script"
$jsonPath = "c:\test\helloWorld.json"
Invoke-RestMethod -Uri $uri -Method POST -Headers @{"Authorization"="Basic $cred"} -ContentType
"application/json" -InFile $jsonPath
```

List Scripts

```
GET /service/rest/v1/script
```

This endpoint allows us to get a listing of all the scripts currently defined in the repository manager.

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/script
```

Building on the example from above, we only have the one script that we just created:

Example Response

```
[ {
  "name" : "helloWorld",
  "content" : "log.info('Hello, World!')",
  "type" : "groovy"
} ]
```

Get Script

```
GET /service/rest/v1/script/{name}
```

This endpoint allows us to get the details for an individual script by its name.

Let's get the details for the helloWorld script that we defined earlier:

```
curl -u admin:admin123 -X GET http://localhost:8081/service/rest/v1/script/helloWorld
```

Example Response

```
{
  "name" : "helloWorld",
  "content" : "log.info('Hello, World!')",
  "type" : "groovy"
}
```

Update Script

```
PUT /service/rest/v1/script/{name}
```

This endpoint allows us to update the contents of an existing script.

Let's update our script to log "Hello, Nexus!" instead of "Hello, World!":

```
curl -u admin:admin123 -X PUT --header 'Content-Type: application/json' \
 http://localhost:8081/service/rest/v1/script/helloWorld \
 -d @helloNexus.json
```

helloNexus.json

```
{
  "name": "helloWorld",
  "content": "log.info('Hello, Nexus!')",
  "type": "groovy"
}
```

Run Script

```
POST /service/rest/v1/script/{name}/run
```

This endpoint allows us to run a script that we have already defined in the repository manager.

Let's run our helloWorld script:

```
curl -u admin:admin123 -X POST --header 'Content-Type: text/plain' \
 http://localhost:8081/service/rest/v1/script/helloWorld/run
```

The JSON response tells us which script ran and the returned result from the script:

Example Response

```
{
  "name" : "helloWorld",
  "result" : "null"
}
```

We should also see something like the following in the console log:

```
2018-01-22 16:28:55,495+0000 INFO  [qtp74765809-267] admin
org.sonatype.nexus.script.plugin.internal.rest.ScriptResource$$EnhancerByGuice$$99ce415 - Hello, Nexus!
```

✓ **Privileges**

To run a script, a user needs to have the nx-script-${script-name}-run privilege. A user with the nx-script-*-run privilege can run all scripts. Note that the will also need privileges assigned to them which allow them to perform the actions that the script does.

Delete Script

```
DELETE /service/rest/v1/script/{name}
```

This enpoint allows us to delete scripts once they are no longer needed.

Let's delete our helloWorld script:

```
curl -u admin:admin123 -X DELETE http://localhost:8081/service/rest/v1/script/helloWorld
```

A successfull deletion returns no content:

Example Response

```
HTTP/1.1 204 No Content
Date: Mon, 22 Jan 2018 22:58:03 GMT
...
```

Writing Scripts

The scripting language used on the repository manager is Groovy[19]. Any editor can be used to author the scripts.

The available APIs are contained in a number of JAR files. All these files, including JavaDoc and Sources archives, are available from the Central Repository[20]. They can be manually downloaded and extracted. E.g. the different versions and the specific JAR files for `org.sonatype.nexus:nexus-core` are available in versioned directories at `http://repo1.maven.org/maven2/org/sonatype/nexus/nexus-core/`.

This manual process can be simplified and improved by the usage of a Maven project declaring the relevant components as dependencies. An example project with this setup called `nexus-script-example` and a few scripts are available in the example project[21].

Maven Project pom.xml Declaring the API Dependencies for Scripting

[19] http://www.groovy-lang.org/

[20] http://search.maven.org/

[21] https://github.com/sonatype/nexus-book-examples/tree/nexus-3.x

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example.automation</groupId>
  <artifactId>nexus-script-demo</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <nx-version>3.3.0-01</nx-version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.sonatype.nexus</groupId>
      <artifactId>nexus-core</artifactId>
      <version>${nx-version}</version>
    </dependency>
    <dependency>
      <groupId>org.sonatype.nexus</groupId>
      <artifactId>nexus-script</artifactId>
      <version>${nx-version}</version>
    </dependency>
    <dependency>
      <groupId>org.sonatype.nexus</groupId>
      <artifactId>nexus-security</artifactId>
      <version>${nx-version}</version>
    </dependency>
    <dependency>
      <groupId>org.sonatype.nexus.plugins</groupId>
      <artifactId>nexus-script-plugin</artifactId>
      <version>${nx-version}</version>
    </dependency>
  </dependencies>
</project>
```

Development environments such as IntelliJ IDEA or Eclipse IDE can download the relevant JavaDoc and Sources JAR files to ease your development. Typically you would create your scripts in `src/main/groovy` or `src/main/scripts`.

The scripting API exposes specific tooling for IntelliJ IDEA that allows you to get access to code completion and similar convenience features, while writing your scripts in this Maven project. Currently the API exposes four main providers with numerous convenient methods:

- `core`
- `repository`
- `blobStore`
- `security`

The API is deliberately designed to be simple to use. It encapsulates complex configuration in single method invocations. Many of the included methods use default values that can be omitted. For example, the method to create a hosted repository using the Maven format in the simplest usage simply requires a name.

```
repository.createMavenHosted("private")
```

This method simply uses the default values for the rest of the parameters and is therefore equivalent to:

```
repository.createMavenHosted("private", BlobStoreManager.DEFAULT_BLOBSTORE_NAME, true,
VersionPolicy.RELEASE, WritePolicy.ALLOW_ONCE, LayoutPolicy.STRICT)
```

You can inspect the default values in the API documentation available by inspecting the declaration of the specific methods in your IDE or by viewing the JavaDoc.

In terms of overall complexity of the scripts created, it is best to break large tasks up into multiple scripts and therefore invocations.

Managing and Running Scripts

Once you have completed the creation of your script, you need to publish it to the repository manager for execution. This is done by REST API invocations.

For Repository Manager version 3.8.x and onwards, the REST API endpoint is:

```
http://localhost:8081/service/rest/v1/script
```

For Repository Manager versions pre 3.8.x, the REST API endpoint is:

```
http://localhost:8081/service/siesta/rest/v1/script
```

ⓘ **Note**

The following examples given below are against NXRM version 3.8+ so please amend the endpoint accordingly if running any NXRM 3 version before that.

This endpoint accepts JSON-formatted payloads with your script as the `content`.

Example JSON formatted file `maven.json` with a simple repository creation script:

```
{
  "name": "maven",
  "type": "groovy",
  "content": "repository.createMavenHosted('private')"
}
```

The JSON file `maven.json` located in the current directory can be published to the repository manager with an HTTP POST like:

```
curl -v -X POST -u admin:admin123 --header "Content-Type: application/json" 'http://localhost:8081/service/
rest/v1/script' -d @maven.json
```

A list of scripts stored on the repository manager can be accessed with:

```
curl -v -X GET -u admin:admin123 'http://localhost:8081/service/rest/v1/script'
```

The same call with a script name appended returns the actual script content.

A script can be executed by sending a POST to the run method of the specific script.

```
curl -v -X POST -u admin:admin123 --header "Content-Type: text/plain" 'http://localhost:8081/service/rest/
v1/script/maven/run'
```

A successful execution should result in a `HTTP/1.1 200 OK`.

Scripts can be removed with a HTTP DELETE operation to the specific script:

```
curl -v -X DELETE -u admin:admin123 'http://localhost:8081/service/rest/v1/script/maven'
```

Parameters

Scripts can receive run-time parameters via the REST API:

```
curl -v -X POST -u admin:admin123 --header "Content-Type: text/plain" 'http://localhost:8081/service/rest/
v1/script/updateAnonymousAccess/run' -d 'false'
```

and receive them as arguments that have to be parsed by the script as desired:

```
security.setAnonymousAccess(Boolean.valueOf(args))
```

Multiple parameters can be passed in via a JSON file in the POST request:

params.json:

```
{
"repoName":"snapshots",
"groupId":"org.some.project",
"artifactId":"someartifact"
}
```

Then the command would be similar to the following:

```
curl -k -u user:pass -X POST --header 'Content-Type: text/plain' http://localhost:8081/service/rest/v1/
script/rebuild-maven-metadata/run -d @params.json
```

And you can access the parameters in your script code:

```
def request = new JsonSlurper().parseText(args);
assert request.repoName: 'repoName parameter is required';
assert request.groupId: 'groupId parameter is required';
assert request.artifactId: 'artifactId parameter is required';
```

Interaction with the REST API for scripts can be done with any scripting language capable of HTTP calls as mentioned above. In the following section you can find some further detailed examples.

Examples

The API for scripts is capable of a number of different tasks. This section provides examples for script writing, publishing and executing them.

The **`simple-shell-example`** project in the scripting section of the example project[22] includes a number of JSON file with simple scripts:

`maven.json`

simplest script to create a hosted Maven repository

`npm.json`

simple script to create a hosted and proxy repository as well as a repository group for npm usage

`bower.json`

[22] https://github.com/sonatype-nexus-community/nexus-scripting-examples

simple script to create a hosted and proxy repository as well as a repository group for bower usage

`anonymous.json`

parameterized script to enable or disable anonymous access

Simple shell scripts are added to contain the curl invocations to manage scripts via the REST API:

`create.sh`

Upload a specified JSON file

`delete.sh`

Delete a script specified by its name

`list.sh`

List all deployed scripts

`run.sh`

Run a script specified by its name

`setAnonymous.sh`

Run the anonymous script on the server with the parameter true or false

`update.sh`

Update an existing script by specifying the name and the JSON file to use for the update

An example sequence of creating and running a script is:

```
./create.sh maven.json
./run.sh maven
```

Subsequently you could list all scripts and delete the maven script with:

```
./list.sh
./delete.sh maven
```

Since scripts are typically longer than a single line and creating them in a separate file in the IDE is recommended, using a helper script that formats a .groovy file into a JSON file and submits it to the repository manager can be a convenient approach.

The **`complex-script`** project in the scripting section of example project[23] includes an example implementation using Groovy invoked from a shell script. All scripts in this folder can be published and executed via the `provision.sh` file. This results in the download of all required dependencies and the upload and execution of the referenced script. Alternatively, you can provision the scripts individually:

[23] https://github.com/sonatype-nexus-community/nexus-scripting-examples

```
groovy addUpdateScript.groovy -u "admin" -p "admin123" -n "raw" -f "rawRepositories.groovy" -h "http://localhost:8081"
curl -v -X POST -u admin:admin123 --header "Content-Type: text/plain" "http://localhost:8081/service/rest/v1/script/raw/run"
```

The following scripts are available:

`npmAndBowerRepositories.groovy`

for NPM and Bower repositories suitable for server-side and client JavaScript-based development

`rawRepositories.groovy`

creates a new blob store and uses it for a hosted raw repository

`security.groovy`

disables anonymous access, creates a new administrator account, creates a new role with a simple expansion to anonymous user role and a user, creates a new role with publishing access to all repositories and a user

`core.groovy`

configures the base URL capability and a proxy server

Logging from your scripts into the repository manager logs is automatically available and performed with the usual calls:

```
log.info('User jane.doe created')
```

The result of the last script line is by default returned as a string. This can be a message as simple as `Success!` or more complex structured data.

For instance, you can easily return JSON using built-in Groovy classes like:

```
return groovy.json.JsonOutput.toJson([result: 'Success!'])
```

which looks like:

```
{
    "result": "Success!"
}
```

Passing parameters to the script can use JSON encoded arguments like:

```
{
  "id": "foo",
  "name": "bar",
  "description": "baz",
  "privilegeIds": ["nx-all"],
  "roleIds": ["nx-admin"]
}
```

which in turn can be parsed using the `JsonSlurper` class in the script:

```
import groovy.json.JsonSlurper

//expects json string with appropriate content to be passed in
def role = new JsonSlurper().parseText(args)

security.addRole(role.id, role.name, role.description, role.privilegeIds, role.roleIds)
```

You can read more about how to work with XML and JSON with Groovy on http://groovy-lang.org/processing-xml.html and http://groovy-lang.org/json.html.

Status API

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

Introduction

This set of endpoints is used to determine the status of an instance. This is useful when conducting some maintenance activities.

Endpoints

Status

```
GET /service/rest/v1/status
```

This endpoint allows us to query if the instance can successfully serve read requests.

Curl Example

```
curl -X GET http://localhost:8081/service/rest/v1/status
```

The response code contains the information. A 200 indicates the instance can serve read requests, a 503 otherwise.

Writable

```
GET /service/rest/v1/status/writable
```

This endpoint allows us to query if the instance can successfully serve write and read requests.

Curl Example

```
curl -X GET http://localhost:8081/service/rest/v1/status/writable
```

The response code contains the information. A 200 indicates the instance can serve write and read requests, a 503 otherwise.

Support API

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

Introduction

This endpoint allows us to download a support zip from an instance.

Endpoints

Supportzip

```
GET /service/rest/v1/support/supportzip
```

This endpoint allows us to download a support zip from an instance.

Curl Example

```
curl -u admin:admin123 -X POST "http://localhost:8081/service/rest/v1/support/supportzip" -H "accept:
application/octet-stream" -H "Content-Type: application/json" -d "{ \"systemInformation\": true,
\"threadDump\": true, \"metrics\": true, \"configuration\": true, \"security\": true, \"log\": true,
\"taskLog\": true, \"auditLog\": true, \"jmx\": true, \"limitFileSizes\": true, \"limitZipSize\": true}" --
output supportzip.zip
```

The response body contains the support zip.

Tasks API

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

Introduction

This set of endpoints allows us to interact with tasks that have been created in the repository manager administrative UI.

Endpoints

List Tasks

```
GET /service/v1/tasks
```

This endpoint allows us to iterate through a listing of all the existing tasks.

Let's get a listing of the current tasks:

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/tasks'
```

Example Response

```
{
  "items" : [ {
    "id" : "3f0173a8-c379-44c3-8ba3-2f0d0290bbfa",
    "name" : "Rebuild all browse trees",
    "type" : "create.browse.nodes",
    "message" : null,
    "currentState" : "WAITING",
    "lastRunResult" : null,
    "nextRun" : null,
    "lastRun" : null
  }, {
    "id" : "dbb1f322-907a-4424-ad82-8d0a89deabe1",
    "name" : "Rebuild all search indices",
    "type" : "repository.rebuild-index",
    "message" : null,
    "currentState" : "WAITING",
    "lastRunResult" : null,
    "nextRun" : null,
    "lastRun" : null
  }, {
    "id" : "07fa0c5d-8217-45da-91d3-fdfe9452e21f",
    "name" : "Publish all maven indices",
    "type" : "repository.maven.publish-dotindex",
    "message" : null,
    "currentState" : "WAITING",
    "lastRunResult" : null,
    "nextRun" : null,
    "lastRun" : null
  }, {
    "id" : "2c8fd1f8-2395-4576-a172-a68089bb2ef7",
    "name" : "Compact default blob store",
    "type" : "blobstore.compact",
    "message" : null,
    "currentState" : "WAITING",
    "lastRunResult" : null,
    "nextRun" : null,
    "lastRun" : null
  }, {
    "id" : "0261aed9-9f29-447b-8794-f21693b1f9ac",
    "name" : "Hello World",
    "type" : "script",
    "message" : null,
    "currentState" : "WAITING",
    "lastRunResult" : null,
    "nextRun" : null,
    "lastRun" : null
  } ],
  "continuationToken" : null
}
```

This endpoint uses a pagination[24] strategy that can be used to iterate through all the tasks if desired.

In this case, we only have one page of results as signified by the `null continuationToken`.

Get Task

```
GET /service/v1/tasks/{id}
```

This endpoint allows us to get the details about in individual task.

Let's take a look at the details for the "Hello World" task (i.e. `id=0261aed9-9f29-447b-8794-f21693b1f9ac`):

```
curl -u admin:admin123 -X GET 'http://localhost:8081/service/rest/v1/tasks/0261aed9-9f29-447b-8794-
f21693b1f9ac'
```

Example Response

```
{
  "id" : "0261aed9-9f29-447b-8794-f21693b1f9ac",
  "name" : "Hello World",
  "type" : "script",
  "message" : null,
  "currentState" : "WAITING",
  "lastRunResult" : null,
  "nextRun" : null,
  "lastRun" : null
}
```

Run Task

```
POST /service/v1/tasks/{id}/run
```

This endpoint allows us to run an individual task.

For example, to run the "Hello World" task from above:

```
curl -u admin:admin123 -X POST 'http://localhost:8081/service/rest/v1/tasks/0261aed9-9f29-447b-8794-
f21693b1f9ac/run'
```

[24] https://help.sonatype.com/display/NXRM3M/Pagination

Example Response

```
HTTP/1.1 204 No Content
Date: Mon, 22 Jan 2018 22:19:47 GMT
...
```

Stop Task

```
POST /service/v1/tasks/{id}/stop
```

This endpoint allows us to stop an individual task. This is the equivalent of cancelling a task in the repository manager UI. Note that not all tasks will respond to a cancellation request.

For example to stop the "Hello World" task:

```
curl -u admin:admin123 -X POST 'http://localhost:8081/service/rest/v1/tasks/0261aed9-9f29-447b-8794-
f21693b1f9ac/stop'
```

In this example the "Hello World" task was not running when the stop request was made. We will get a 409 response code that signifies that the requested task was not currently running:

Example Response

```
HTTP/1.1 409 Conflict
Content-Length: 0
Date: Mon, 22 Jan 2018 22:22:33 GMT
...
```

Pagination

Many of the REST API's make use of a pagination strategy for dealing with operations that can return a large number of items. This strategy is built around the notion of a `continuationToken` and a page size that determines the maximum number of items that can be returned in a single response.

When a `continuationToken` is present in a response and has a non-null value, this signifies that there are more items available:

Example Request

```
GET /service/rest/v1/<api>?<query>
```

Example Response

```
{
  "items" : [
    ...
  ],
  "continuationToken" : "88491cd1d185dd136f143f20c4e7d50c"
}
```

The API that produced the response will take a `continuationToken` as an additional argument to the original query to signify that the next page of results is desired:

Example Request (next page)

```
GET /service/rest/v1/<api>?<query>&continuationToken=88491cd1d185dd136f143f20c4e7d50c
```

If this response also contains a non-null `continuationToken`, then its value can again be added to the original query to get the next page. This continues until the response returns a `continuationToken` with a value of `null` which signifies that there are no more pages of results.