JFrog XRAY

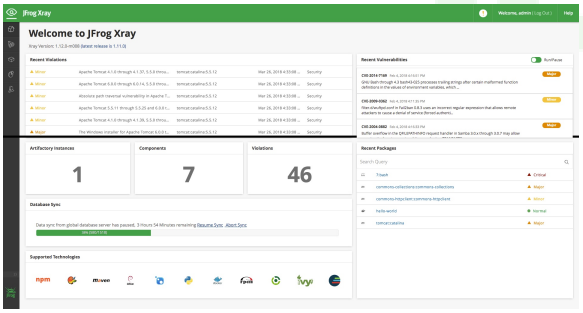
Version 1.0 User Guide

Welcome to JFrog Xray

Overview

JFrog Xray works with JFrog Artifactory to perform universal analysis of binary software components at any stage of the application lifecycle providing radical transparency that leads to trust in your software. By scanning binary components and their metadata, recursively going through dependencies at any level, JFrog Xray provides unprecedented visibility into issues lurking in components anywhere in your organization. Xray's interface with Artifactory gives it the exclusive advantage of combining any number of data feeds with the exhaustive metadata stored within Artifactory to detect different issues without needing access to source code. JFrog Xray is also fully automated through a rich REST API that lets it integrate with a CI/CD pipeline and allows other binary analysis tools to build on its unique capabilities.

- **Universal**
  In line with JFrog's universal approach, Xray supports a variety of package types supported by Artifactory.
- **Open for integration**
  While Xray comes with its own database of software components and vulnerabilities out-of-the-box, it is also open to integration with other databases and tools. Xray comes built-in with integration to tools such as Whitesource, Aqua and Blackduck hub. In addition, using Xray's open API, customers can integrate Xray with their own systems and data feeds.
- **Open for different issue types**
  Xray is not limited to security vulnerabilities; it can receive any type of information about software component that can help you make decisions. For example, you can provide Xray with information about components that have performance issues or severe defects and the impact that these components have on your software.
- **Deep scanning**
  Xray performs a deep scan of artifacts, recursively going through dependencies at any level and creating a graph of relationships between software components. For example, when analyzing a Docker image, if Xray finds that it contains a Java application it will also analyze all the .jar files used in this application.
- **Impact analysis**
  Xray analyzes how an issue in one component affects all others in your company and displays the chain of impact in a component graph.
- **Native integration with Artifactory**
  Xray is the only tool that is natively integrated with JFrog Artifactory.



Once you have installed Xray, to start analyzing your repositories and reveal vulnerabilities in your system, please refer to Configuring Xray.

How Does JFrog Xray Protect You

JFrog Xray is the only product that takes a dual approach to protecting you against issues using a unique combination of:

Deep Recursive Scanning

JFrog Xray recursively scans components in your system, recursively drilling down to analyze even the smallest binary component that affects your software.

Continuous Impact Analysis

Page Contents

Quick Links

Release Notes	Xray RES

Read More

JFrog Xray continuously scans and analyzes existing components, even those long since deployed to production, and provides alerts and notifications for just-discovered vulnerabilities.

Custom API-Driven Automation

Through an open REST API, JFrog Xray lets you define a custom regimen of automated analysis for all components in your system.

The Xray-Artifactory Edge

As a complementary product to JFrog Artifactory, JFrog Xray has access to the wealth of metadata Artifactory stores which, combined with deep recursive scanning, puts Xray in a unique position to analyze the relationships between binary artifacts and provide radical transparency into your component architecture to reveal the impact that a vulnerability in one component has on any other.

PDF Download

The Xray User Guide is available for download in PDF format. Click this link to download the latest version:

Note that the online version may be more up-to-date.

Installing Xray

Overview

JFrog Xray is a complementary product to JFrog Artifactory and is run as a separate installation as a set of microservices.

Both Docker and Non-Docker installation flavours are as quick and easy as possible, you only need to download a simple script that manages download and installation of all the other components needed to run Xray.

To get started, make sure your system complies with the requirements in the following section before you proceed to download and install Xray.

> **Installer Log**
>
> As part of the installation/upgrade Xray creates a log file to track the installation process. Each installation/upgrade will create a new install log file with the following format:
>
> ${INSTALLER_DIR}/${SCRIPT_NAME}.${DATE}.log

> **Xray High Availability**
>
> This user guide is for JFrog Xray 1.x.
>
> Xray HA is available from version 2.0. To learn more, please refer to Xray High Availability in the JFrog Xray 2.x User Guide

> **Using non-interactive automated scripts to install Xray**
>
> To install/upgrade Xray using automation, add the following to your environment variables and the `xray-env.conf` file:
>
> `USE_DEFAULTS=true`
>
> Existing parameters will be used in the automation process.

System Requirements

Hardware

JFrog Xray requires the following hardware:

- Processor: 8 cores
- RAM Memory: 16 GB
- Storage: 100 GB
- An Artifactory separate host machine

> **Allocated storage space may vary**
>
> Xray downloads and then deletes fetched artifacts after indexing. However, in order to have more parallel indexing processes, and thereby more temporary files at the same time would require more space.
>
> This is especially applicable for large BLOBs such as Docker images.

Platforms

JFrog Xray supports any non-Windows platform that can run Docker v1.11 and above, and in addition, has been tested and verified to run as a non-Docker installation on the following 64-bit flavors of Linux:

- Ubuntu 14.04
- Centos 7.x
- Debian 8.x
- Red Hat 6.x

Page Contents

- Red Hat 7.x

File Handle Allocation Limit

> **Avoid performance bottlenecks**
>
> In the process of deep recursive scan in which Xray indexes artifacts and their dependencies (metadata), Xray needs to concurrently manage many open files. The default maximum number of files that can be opened concurrently on Linux systems is usually too low for the indexing process and can therefore cause a performance bottleneck. For optimal performance, we recommend increasing the number of files that can be opened concurrently to 100,000 (or the maximum your system can handle) by following the steps below.

Use the following command to determine the current file handle allocation limit:

```
cat /proc/sys/fs/file-max
```

Then, set the following parameters in your *`/etc/security/limits.conf`* file to the **lower** of 100,000 or the file handle allocation limit determined above.

The example shows how the relevant parameters in the *`/etc/security/limits.conf`* file are set to 100000. The actual setting for your installation may be different depending file handle allocation limit in your system.

```
root hard nofile 100000
root soft nofile 100000
xray hard nofile 100000
xray soft nofile 100000
postgres hard nofile 100000
postgres soft nofile 100000
mongodb hard nofile 100000
mongodb soft nofile 100000
```

Screencast

Docker

JFrog Xray requires Docker v 1.11 and up to be installed on the machine on which you want to run Xray. For instructions on installing Docker, please refer to the Docker documentation.

Browsers

Xray has been tested with the latest versions (known at the time of release) of Google Chrome, Firefox, Internet Explorer, Microsoft Edge and Safari.

Artifactory

From **version 1.1**, JFrog Xray supports **JFrog Artifactory v4.0** and above.

Older versions of JFrog Xray only support JFrog Artifactory v4.11 and above.

> **Recommended Artifactory Version**
>
> We recommend using JFrog Xray with **JFrog Artifactory v4.12** and above for best integration and performance experience.

JFrog Xray 1.12 was co-released with Artifactory 5.10. Due to a fundamental change in the integration of Xray with Artifactory in these versions, the following matrix describes version compatibility going forward:

		Xray Version	
		1.12+	**<1.12**
Artifactory Version	**5.10 +**	✅ Since both Artifactory and Xray are upgraded, the new integration is fully functional as designed.	⚠️ This combination is supported. Artifactory will continue to display each artifact's scan status, however, it will use previous mechanism that uses properties.

	<5.10	In this combination, the integration will not work since an older version of Artifactory does not query Xray for scan status, and the new version of Xray does not attach properties to an artifact.	If neither Artifactory nor Xray are upgraded, the integration will work using the previous mechanism that displayed scan status as a set of properties on the artifact.

Feature Compatibility

Artifactory and Xray progress independently, and some features in Xray require specific versions in Artifactory for support as described in the following table:

Feature	Artifactory Version	Xray Version
CI/CD Integration	v >= 4.16	v >=1.6
Bi-directional connection test	v >= 4.15	v >=1.3
Xray license validation	v >= 4.11	v >=1.0
Download blocking based on Xray alerts	v >= 4.13	v >=1.1
Xray section in General Information tab of selected artifact in Artifactory's tree browser	v >= 4.11	v >=1.0
Synchronizing artifacts via REST API	v >= 4.11	v >=1.0
Synchronizing artifacts through a user plugin	4.11 > v >=4.0	v>=1.1

Supported Technologies

JFrog Xray supports scanning and impact analysis for the following package formats:

- Java (Maven, Gradle, Ivy)
- JavaScript (NPM, Bower)
- .NET (Nuget)
- Python (PyPi)
- Docker
- Debian
- RPM
- SBT

Download and Installation

JFrog Xray may be installed as a Docker image, or as a non-Docker installation for each of the supported flavors of Linux. Once you have downloaded your preferred installer, follow the installation instructions in the corresponding sections below.

The **Xray Download Page** provides the JFrog Xray installer for any of the supported platforms (Docker or Linux flavors).

> **Keep Xray on your $PATH**
>
> Make sure to save the downloaded file in one of the locations defined in your $PATH environment variable so it is accessible from anywhere on your machine.

Docker Installation

> **Running Xray without Docker**
>
> To run Xray as a non-Docker installation, please refer to Linux Installations.

The JFrog Xray Docker image may be installed on any platform supporting Docker v1.11 and above. To install Xray as a Docker image, make sure you have an network connection and follow the instructions below:

1. **Make `xray` executable**
   To give xray execute privileges on your machine, run:

```
chmod +x xray
```

2. **Install and start Xray**
   The installation process will prompt you for a "root folder". You may keep the defaut (current) location or specify another location on your machine. Choose this location carefully since you may not change it later, and this is where JFrog Xray saves its data, configuration files and logs. The Xray installer will only prompt you for this location for initial installation. It is stored for later use when upgrading.

   To install Xray, run the following command:

   ```
   sudo ./xray install
   ```

   Using External Databases

   JFrog Xray uses several databases for different features of its operation. Until version 1.10, Xray installed an instance of all of these databases dedicated for its own use.

   From version 1.10, Xray gives you the option of using your own **MongoDB** and **Postgres** databases if you have these already installed and in use in your organization.

   For more details, please refer to Using External Databases.

   To start Xray, run the following command:

   ```
   ./xray start
   ```

Port Configuration

Make sure ports on your JFrog Xray and JFrog Artifactory installations are properly configured to enable communication between the two applications.

Upgrading on Docker

For instructions on how to upgrade an existing installation, please refer to Upgrading Xray.

Interacting with the Docker Installer

In addition to managing installation, the xray installation script can provide additional information or perform additional tasks on your installation such as restarting Xray, displaying log files and more. For details, run:

```
./xray help
```

Linux Installation

Using RPM?

For an RPM installation, please ensure the following conditions hold:

- JFrog Xray must be installed on a different machine from JFrog Artifactory.
- The umask (user file creation mode mask) must have a default setting of 0022, 022, 0002 or 002

Using a third-party log collector

To use an external log collector that requires a separate user for Xray (e.g. Sumologic, Splunk) , you can adjust the permissions on the *${XRAY_DATA}/logs* folder to allow the the log collection service to perform read operations on the generated log files as follows:

1. Add the log collection service user to the relevant group if needed (the user and group that installed and started Xray)
2. Apply the user and group permissions as needed on the *${XRAY_DATA}/logs* directory using:

```
$ chmod -R 640 ${XRAY_DATA}/logs
```

3. Adjust the group read inheritance permissions setgid bit using:

```
$ chmod -R 2755 ${XRAY_DATA}/logs
```

This will cause the generated log files to inherit the folder's group permissions.

The Xray Linux installation follows standard conventions and installs Xray in the following folders:

Application files	/opt/jfrog/xray
Data files	**Default:** /var/opt/jfrog/xray/data/ The installation script will prompt you for an optional alternative location.
Log files	**${XRAY_DATA}**/logs
Log configuration files	**${XRAY_DATA}**/config
PostgreSQL home directory	**Default:** /var/opt/jfrog/postgres The installation script will prompt you for an optional alternative location.

In all of the instructions below, replace the **<linux-flavor>** place-holder with one of **centos**, **debian**, **ubuntu** or **redhat** according to the flavor of Linux on which you are operating.

The installation instructions for all of the supported flavors of Linux are the same.

1. **Extract the downloaded installation archive**

```
tar -xzf xray-<linux-flavor>-latest.tar.gz
```

2. **Run the installation script**
(if you are not running as "root", prepend the following command with "sudo")

```
./installXray-<linux-flavor>.sh
```

Using External Databases

JFrog Xray uses several databases for different features of its operation. Until version 1.10, Xray installed an instance of all of these databases dedicated for its own use.

From version 1.10, Xray gives you the option of using your own **MongoDB** or **Postgres** databases if you have these already installed and in use in your organization.

For more details, please refer to Using External Databases.

Upgrading on Linux

For instructions on how to upgrade an existing installation, please refer to Upgrading Xray.

Interacting with the Linux Installer

Make sure Xray fully started

Verify all the required Xray components and connected databases are up and running by the following command:

```
./xray.sh status all
```

Use the below command to start all Xray components:

```
./xray.sh start all
```

It is also possible to exclude the '*all*' flag which will make the script run or check only for the running **Xray** services (without the databases):

```
./xray.sh status
./xray.sh start
```

The installation script offers facilities for maintenance. Run the following commands as "root" or prepend them with "sudo".

```
./xray.sh <command> <target (optional)>
```

where:

<command> can take one of the following values:

start	Start the service
stop	Stop the service
restart	Restart the service
status	Display the service status (e.g. running, stopped...)
info	Displays version information for each service
deployServices	Deploy the service (only available for the `xray` service)
removeServices	Remove the service (only available for the `xray` service)

<target> Optional. When omitted, the command only applies to the Xray service.

all	Apply the command to all services

Using External Databases

JFrog Xray uses several databases for different features of its operation. Until version 1.10, Xray installed an instance of all of these databases dedicated for its own use.

From version 1.10, Xray gives you the option of using your own **Postgres** or **MongoDB** databases if you have these already installed and in use in your organization.

Supported database versions

Currently, Xray supports the following external database versions:

PostgreSQL: version 9.5.2

MongoDB: version 3.2.6

It is up to you to choose which, if any of these databases to externalize when you install Xray.

During the installation process, the Xray installation script will prompt you with questions about whether to install an internal database or to use one already installed in your organization. Simply respond to these prompts as required.

> **You take full responsibility for your own databases**
>
> If you choose to have Xray use any of your own databases for its operation, you take full responsibility for the maintenance, backup and correct functioning of these databases.

For example, the Xray installation script will ask if you would like to install **Postgres** or **MongoDB.**

If you respond with a "Y", Xray will install **Postgres** or **MongoDB** for its own use.

```
Would you like to install PostgreSQL instance? [Y/n]: n
Type a PostgreSQL connection string [postgres://xray:xray@postgres:5432/xraydb?sslmode=disable]:
postgres://xray:xray@<MACHINE_IP>:5432/xraydb?sslmode=disable
Would you like to install MongoDB instance? [Y/n]: n
Type a MongoDB connection string [mongodb://xray:password@mongodb:27017/?authSource=xray&authMechanism=SCRAM-
SHA-1]: mongodb://xray:password@<MACHINE_IP>:27017/?authSource=xray&authMechanism=SCRAM-SHA-1
```

Accessing Xray

JFrog Xray can be accessed using the following URL:

`http://<SERVER_NAME>`:8000/web/#/home

For example, if you are accessing Xray on a machine called "myserver" you would use: http://myserver:8000/web/#/home

> **Xray access URL is not its base URL**
>
> Be careful not to confuse Xray's access URL with its **base URL**.
>
> Xray's access URL is: <XRAY_BASE_URL>/web/#/home
>
> If you set the access URL in the **Xray Base URL** field of Xray's basic configuration, connected Artifactory instances will not be able to communicate with Xray

---

Activating Xray

Purchase - Automatic Activation

If you have purchased Xray, it is activated automatically when you connect it to a licensed Artifactory instance - one that has an Xray license incorporated into the Artifactory license.

> **Purchased a license?**
>
> Make sure to activate your Artifactory instances with a comprehensive license that includes Xray activation.

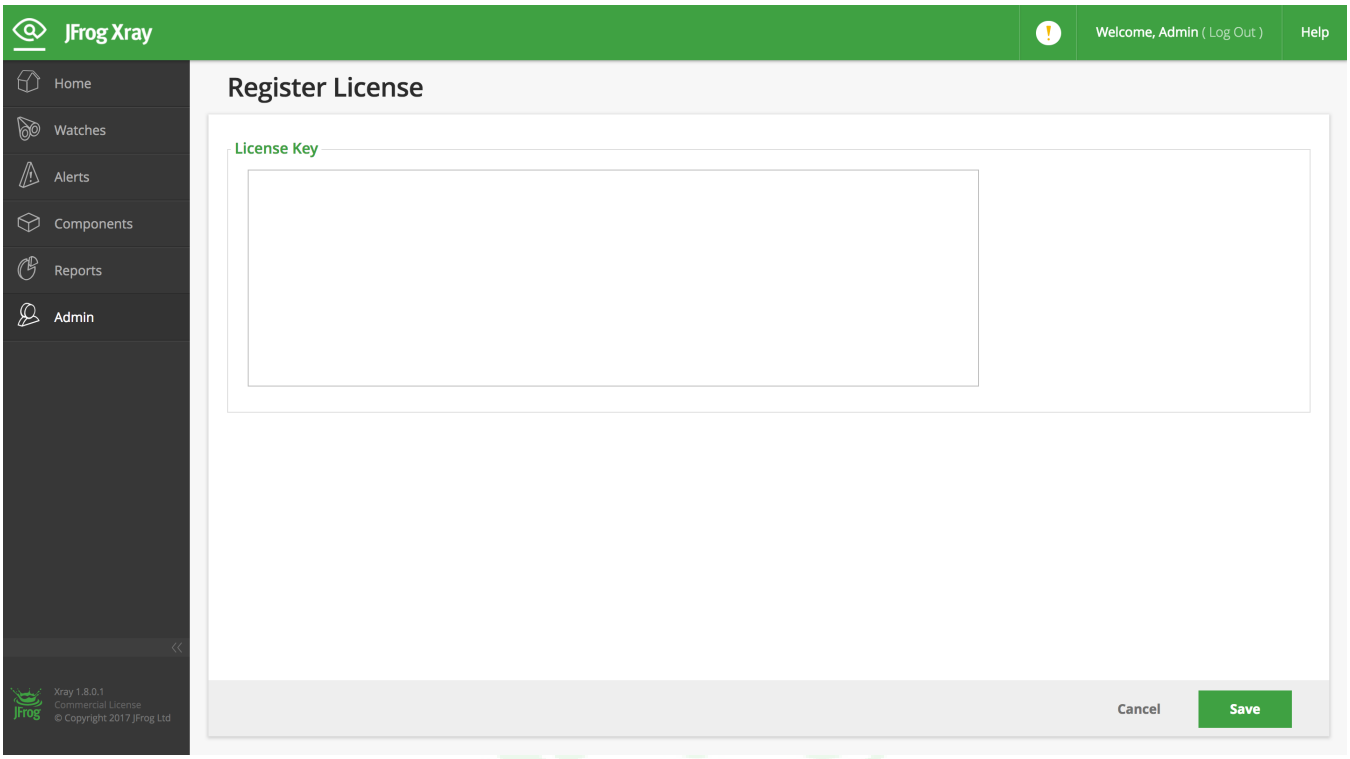
If you are currently evaluating JFrog Xray (i.e. you are on a free trial), you need to set your license manually in order to activate it.

Free Trial - Manual Activation

If you have requested an evaluation of Xray, your license key will be provided to you as part of the registration process

> **Problems activating Xray?**
>
> If you have any problems receiving your license or activating Xray, please contact JFrog Support.

Your administrator should enter the license key manually into the corresponding field in the **Admin** module under **Register License.**



Default Admin User

Once installation is complete, Xray has a default user with admin privileges predefined in the system:

User: admin

Password: password

> **Change the admin password**
>
> We strongly recommend changing the admin password as soon as installation is complete.

Upgrading Xray

Overview

Xray is composed of several micro services; some are core to Xray while others are third party services such as MongoDB or PostgreSQL. One of the micro services is the Xray installer which, in addition to initial installation, is also responsible for the upgrade process allowing you to upgrade frequently without the risk of losing data. As a result, the process of upgrading Xray is very similar to the installation process.

Page contents

Database Sync Required

When upgrading Xray to version 1.8 and above, from a version below 1.8:

- Make sure the DB sync completes successfully
- If you're synchronizing Xray with the Global Database Server in offline mode, to complete the upgrade process, you need to perform a manual **offline database sync** using the **JFrog CLI** version 1.10.0 and above.

Installer Log

As part of the installation/upgrade Xray creates a log file to track the installation process. Each installation/upgrade will create a new install log file with the following format:

${INSTALLER_DIR}/${SCRIPT_NAME}.${DATE}.log

Using non-interactive automated scripts to install Xray

To install/upgrade Xray using automation, add the following to your environment variables and the `xray-env.cnf` file:

```
USE_DEFAULTS=true
```

Existing parameters will be used in the automation process.

Upgrading on Docker

Download the installation script

Upgrading Xray's Docker distribution is done using the latest installation script that you need to download from the **Xray Download Page.**

Using External Databases

JFrog Xray uses several databases for different features of its operation. Until version 1.10, Xray installed an instance of all of these databases dedicated for its own use.

From version 1.10, Xray gives you the option of using your own **MongoDB** or **PostgreSQL** databases if you have these already installed and in use in your organization.

If you wish to use your own external database, you first need to perform a number of preliminary operations before upgrading Xray.

For more details, please refer to Using External Databases.

After downloading the latest installation script from the **Xray Download Page,** to upgrade your Xray installation, follow the steps below:

1. Give the installation script "execute" privileges on your machine.

```
chmod +x xray
```

2. Stop your current Xray instance:

```
xray stop
```

3. Run the `upgrade` command on the installation script

```
xray upgrade
```

4. Start xray

```
xray start
```

> **Using External Databases**
>
> To use an external database, that's already installed in your organization, follow the process described here.

Interacting with Installation Script

In addition to managing installation and upgrade, the installation script can provide additional information or perform additional tasks on your installation such as restarting Xray, displaying log files and more. For details, run:

```
xray help
```

Upgrading on Linux

Xray is supported on a variety of flavors of Linux and follows standard conventions for folder structure. For details, please refer to Linux Installation.

To upgrade Xray running on Linux, follow the instructions below replacing <linux-flavor> with the flavor you are using:

> **Using External Databases**
>
> JFrog Xray uses several databases for different features of its operation. Until version 1.10, Xray installed an instance of all of these databases dedicated for its own use.
>
> From version 1.10, Xray gives you the option of using your own **MongoDB** or **PostgreSQL** databases if you have these already installed and in use in your organization.
>
> If you wish to use your own external database, you first need to perform a number of preliminary operations before upgrading Xray.
>
> For more details, please refer to Using External Databases.

1. Download the latest installation script for your Linux distribution from the Xray Download Page. The installation script is used for a variety of functions, and is also used for the upgrade process
2. Extract the downloaded installation archive to a temporary folder `/tmp`

```
tar -zxvf xray-<linux-flavor>-latest.tar.gz
```

3. Run the installation script located in the extracted folder
   (if you are not running as "root", prepend the following command with "sudo")

```
./installXray-<linux-flavor>.sh
```

> **Using External Databases**
>
> To use an external database, that's already installed in your organization, follow the process described here.

Interacting with Installation Script

The installation script offers several facilities for maintenance. For details, please refer to Interacting with the Linux Installer.

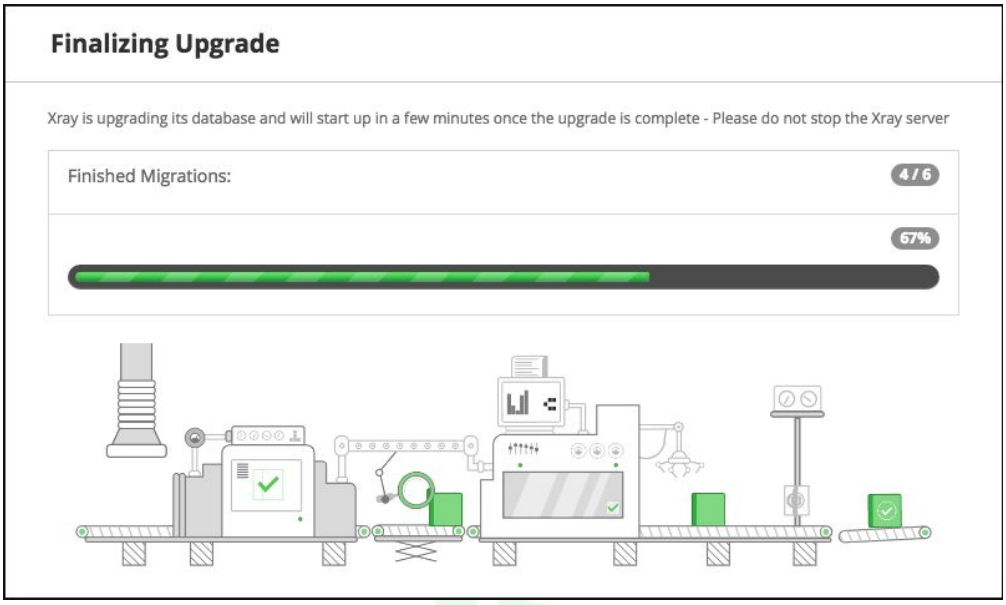
Modifying Default File Locations

When upgrading your Xray installation, the installation script places files in default locations.

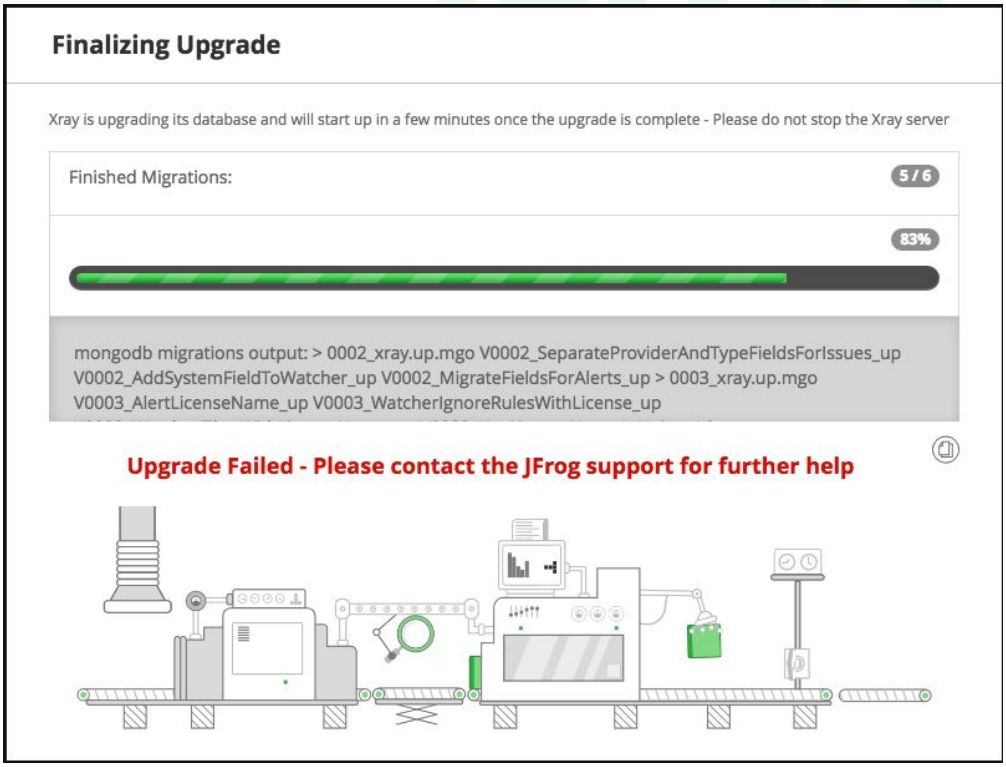
For the Xray data files and PostgreSQL home directory, you may modify the location from the default. For details, please refer to the README file in your Xray home directory.

Upgrading the Database

In some cases, when upgrading Xray to the latest version, Xray also needs to update its database (for example, if the database schema changes). While Xray handles this internal upgrade process automatically, it is a process which may take some time, and it is important that you don't interrupt the upgrade process and let it run to completion. During the upgrade process, Xray will provide a visual indication of its progress.



In case of an issue, Xray will stop the upgrade process and will provide error information which you can share with JFrog support for further help, this can be also troubleshooted.



Using External Databases

JFrog Xray uses several databases for different features of its operation. Until version 1.10, Xray installed an instance of all of these databases dedicated for its own use.

From version 1.10, Xray gives you the option of using your own **MongoDB** or **PostgreSQL** databases if you have these already installed and in use in your organization.

> **Supported versions**
>
> Xray currently supports the following versions for each of these databases:
>
> postgres:9.5.2
> mongo:3.2.6

It is up to you to choose which, if any of these databases to externalize when you upgrade Xray.

> **You take full responsibility for your own databases**
>
> If you choose to have Xray use any of your own databases for its operation, you take full responsibility for the maintenance, monitoring, backup and correct functioning of these databases.

> **Only externalize once**
>
> You only need to follow the process below to use your own database once. Subsequent upgrades can be performed as usual, without having to follow the process below.

To use your own databases with Xray, follow the procedure below:

1. Stop Xray microservices

```
./xray.sh stop
```

2. Back up the Xray configuration file using the following command:

```
mv <XRAY_HOME_FOLDER>/config/xray_config.yaml <XRAY_HOME_FOLDER>/config/xray_config_orig.yaml
```

   During the upgrade process, the default configuration file will be recreated in the same location.
3. Do a DB dump to back up all the databases that you want Xray to use (i.e. those which you are externalizing for Xray).
4. Stop all the external databases that you want Xray to use

```
./xray.sh stop all
```

5. Prepare all the external databases that you want Xray to use by adding the required schema as described in the corresponding snippets below:

Adding the required schema to MongoDB

```
//Creating default admin user
var adminUser = {
    user:"admin",
    pwd: "password",
    roles: ["root"],
    customData: {
        createdBy: "JFrog Xray installer"
    }
}
db.getSiblingDB("admin").createUser(adminUser)

//Creating default xray user
var xrayUser = {
    user:"xray",
    pwd: "password",
    roles: ["dbOwner"],
    customData: {
        createdBy: "JFrog Xray installer"
    }
}

//Authenticating as admin to create xray user
var loginOutput = db.getSiblingDB("admin").auth(adminUser.user,adminUser.pwd)
db.getSiblingDB("xray").createUser(xrayUser)
```

Adding the required schema to PostgreSQL

```
CREATE USER xray WITH PASSWORD 'xray';
CREATE DATABASE xraydb WITH OWNER=xray ENCODING='UTF8';
GRANT ALL PRIVILEGES ON DATABASE xraydb TO xray;
```

6. Purge your current installation of Xray:

```
With Aptitude:
apt-get purge xray

With YUM:
yum erase xray
```

7. Manually clean up your previous installation

> **Manual cleanup required**
>
> Purging your previous installation of Xray only removes Xray's configuration folder found, by default, at `/opt`.
>
> Therefore, after purging your previous version, make sure to clean any residual database files left in your system and the data folder found, by default, at `/var/opt`.

8. Continue with upgrading Xray as described in Upgrading on Docker or Upgrading on Linux according to your installation type.

Getting Started

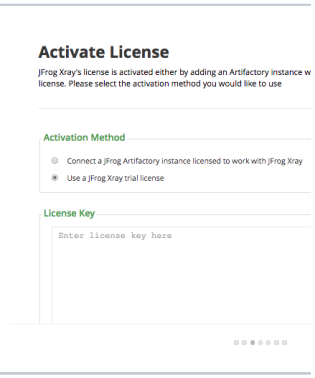
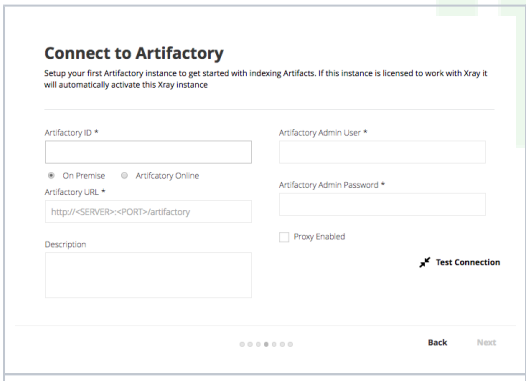
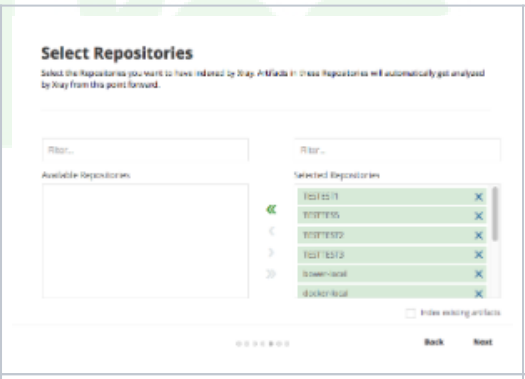
Overview

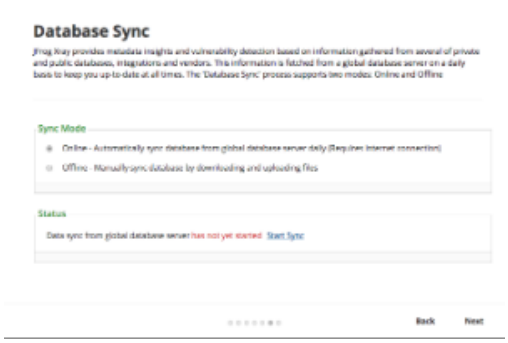
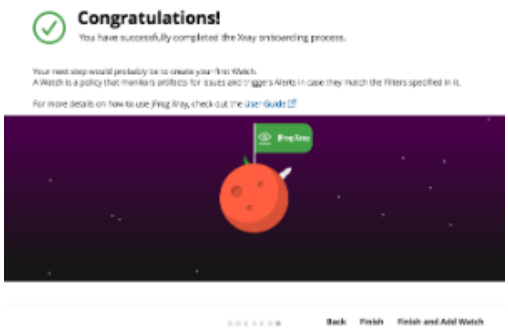
Once you have completed download and installation, it's very easy to get started with Xray by going through the Onboarding Wizard

Page contents

Onboarding Wizard

After installation, first time you run Xray, you will be guided through a short onboarding wizard to make sure you have the minimal parameters needed to run Xray with the following steps:

		
Welcome: The beginning of the onboarding wizard. Click **Next** to get started.	**Basic Configuration:** Define the Xray Base URL through which Xray will be accessed, and configuration for your proxy server if you have one. Note that the Xray Base URL should not be defined as "localhost", or "127.0.0.1", and should not include the " /web" element.	**Activate License:** Activates your connecting to a JFrog Artifactory i work with Xray, or using a JFrog X
		
Connect to Artifactory: Connect to your first Artifactory instance. You can connect to additional instances at any time later on. Once you have configured the connection, you can run a bi-directional connection test to make sure Xray can communicate with Artifactory and vice versa.	**Select Repositories:** Select repositories for indexing in your selected Artifactory instance.	**Select Builds for Index:** Select b your selected Artifactory instance.

	
Database Sync: Synchronize data from the global database server.	Once you have completed the onboarding process, click **Finish** to start using Xray, or **Finish and Add Watch** to create your first Watch. You can update your configuration and add more configuration parameters at any time. For details, please refer to Configuring Xray.

Configuring Xray

Overview

To set up your initial Xray configuration, we recommend that you go through the **Onboarding Wizard** which will be invoked automatically first time you run Xray.

Once you have completed the Onboarding Wizard you may go through the following steps as needed at any time to update and add to your configuration.

1. **Connect to additional Artifactory instances and specify repositories that should be analyzed**
2. **Manage Users**
3. **Configure a mail server**
4. **Configure webhooks**
5. **Define watches**
6. **Indexing artifacts**
7. **Synchronizing the database**

Xray Configuration File

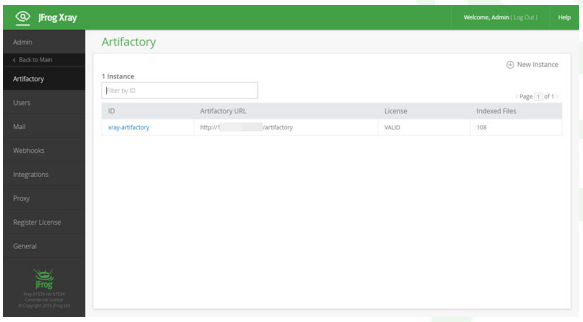
Xray's configuration parameters are stored in its configuration file which is located in:

${XRAY_DATA}/config/xray_config.yaml when running on other supported flavors of Linux.

Connecting to Artifactory

You may connect Xray to several instances of JFrog Artifactory. When a connection between Xray and Artifactory is established, a user with "admin" privileges called **xray** is created in Artifactory which allows Xray to access the data it needs to perform its different analyses and functions.

To view the list of Artifactory instances connected, in the **Admin** module, select **Artifactory**.



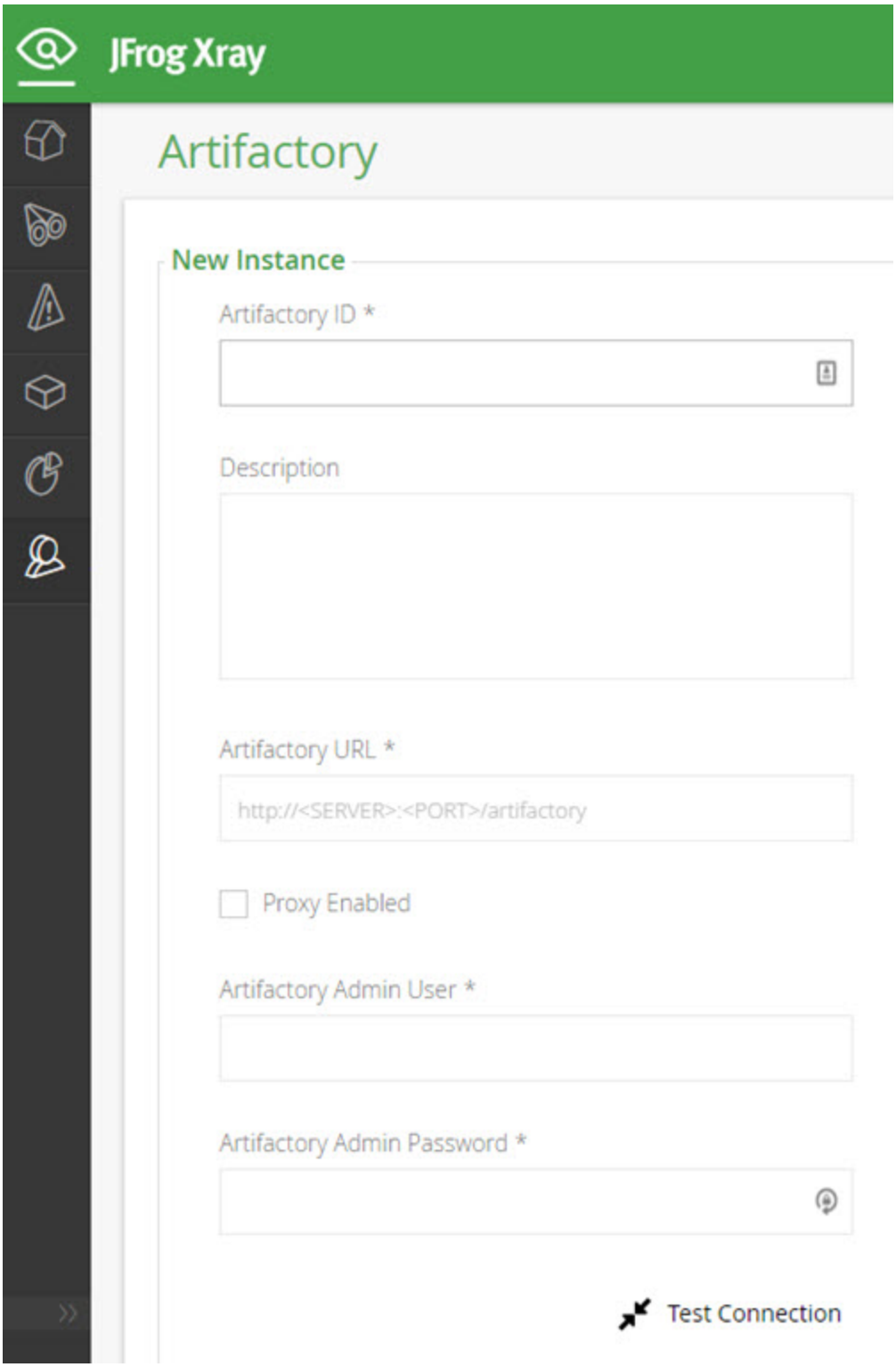
Page Contents

The screen displays the list of Artifactory instances you have added to Xray showing:

ID	The Artifactory ID you provided when registering the instance.
Artifactory URL	The URL of the registered Artifactory instance, including the "/artifactory" path (as in the screenshot).
License	Indicates if the Xray license in the selected Artifactory instance is valid or not.
Number of Indexed Files	Indicates the number of files has indexed for the Artifactory instance according to the repositories specified for analysis in Artifactory

Adding a New Instance

To add a new instance, click the **New instance** link and fill in the form displayed.

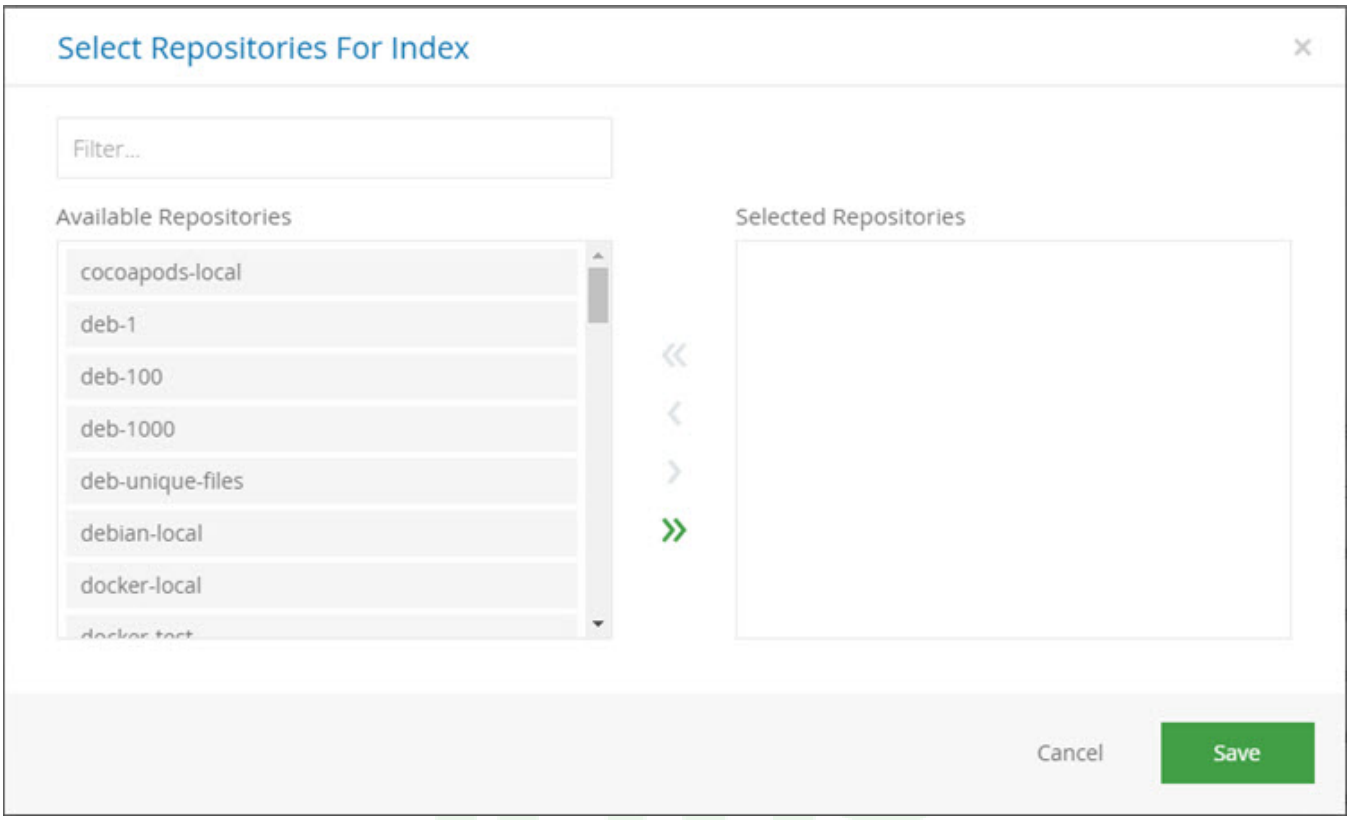


Artifactory ID	A logical identifier for this Artifactory instance.
Description	A description of this instance.
Artifactory URL	The URL by which this instance will be accessed.
Proxy Enabled	When set, Xray will connect to this Artifactory instance through the HTTP proxy defined in the **Admin** module.

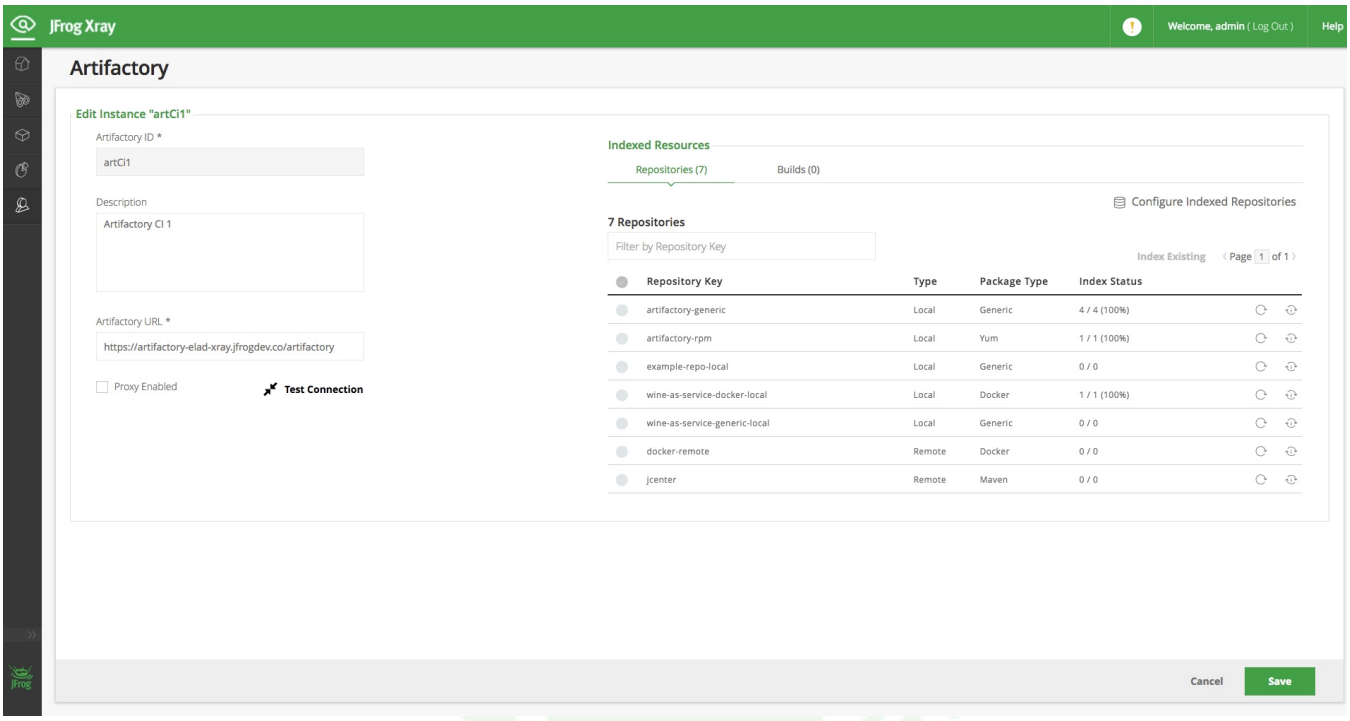
Admin User	A user with admin privileges in this instance.
Admin Password	The admin password.
Test Connection	Tests the connection between Xray and Artifactory (in both directions) to ensure you have configured it correctly.

Specifying Repositories for Analysis

To avoid a lengthy and intensive analysis processes, Xray does not analyze all repositories in the connected Artifactory instance. Instead, after you create a new Artifactory instance, Xray will display the repositories in that instance so you can select which ones should be indexed for analysis.



Once you have specified repositories for analysis in Artifactory, they will appear in the **Repositories** tab under **Indexed Resources** in the Artifactory Instance page (in the **Admin** module under **Artifactory**).



You may now index artifacts in the selected repositories for analysis as described in Indexing Artifacts.

Specifying Builds for Analysis

To avoid a lengthy and intensive analysis processes, Xray does not analyze all builds in the connected Artifactory instance. Instead, after you create a new Artifactory instance, in the **Builds** tab of the **Indexed Resources** panel, you can select which builds should be indexed for analysis.

Managing Users

Xray supports two ways to manage users:

1. Through an Authentication Provider
2. Defining Users Internally

Managing Users Through an Authentication Provider

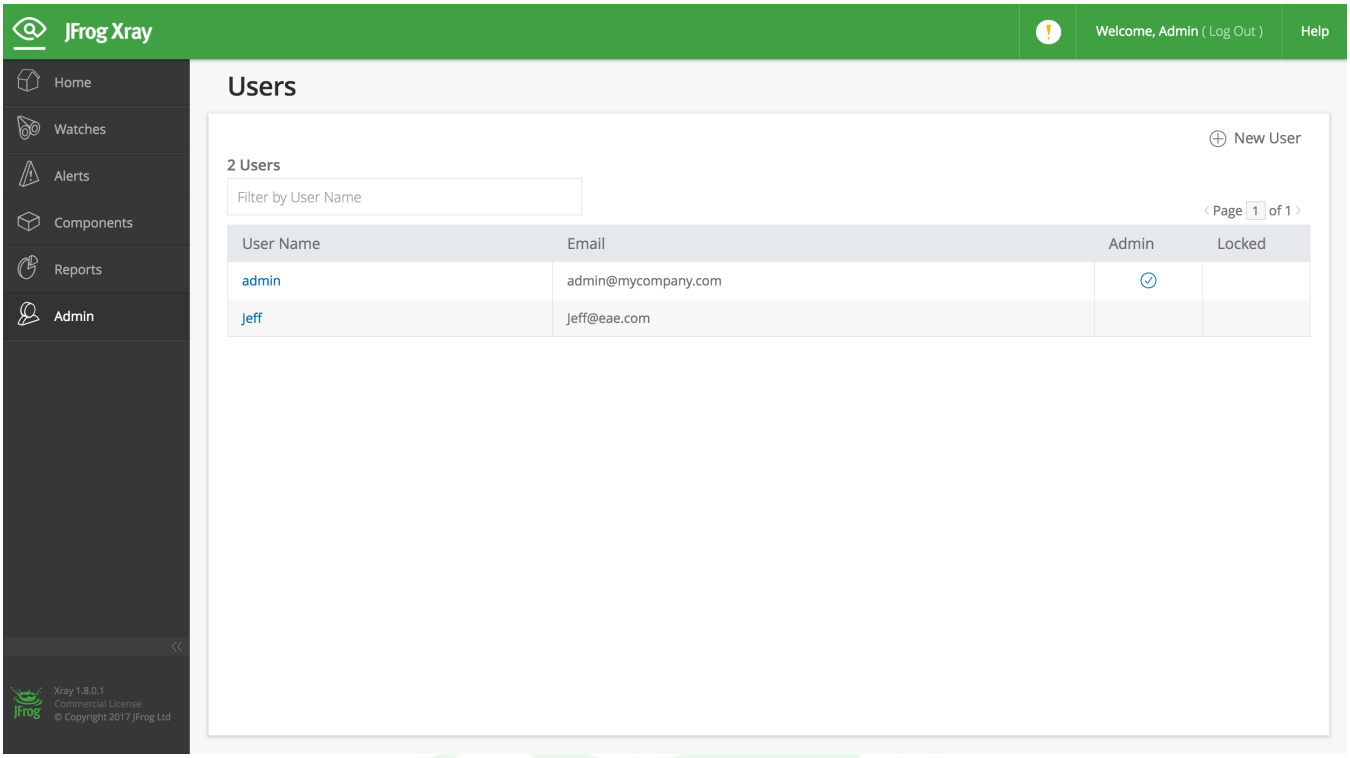
From version 1.9, you can set one of the connected Artifactory instances as an Authentication Provider. In this case, Artifactory will first try to authenticate users through the means set for that instance (whether LDAP/Crowd or SAML).

For details, please refer to Authentication.

Managing Users Internally

Using an Authentication Provider is not mandatory, and you can always define users internally within Xray.

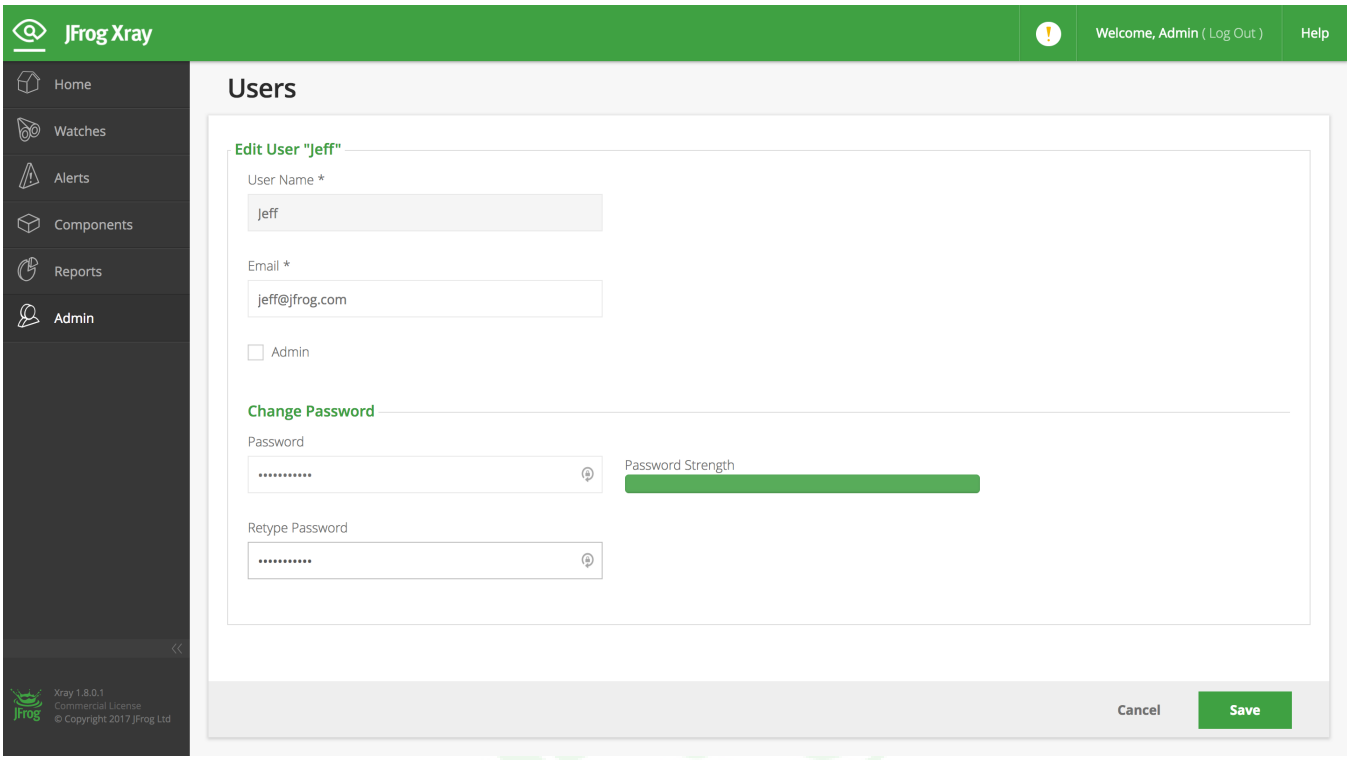
You can view the list of users registered with Xray under the **Admin** module under **Users**.



User Name	The user name with which the user should log on to Xray
Email	The user's email. This is the default email to which notifications will be sent for this user's watchers.
Admin	Indicates if this user has admin privileges

Creating and Editing a User

To add a new user, click **New user,** or click an existing user in the table displayed to edit their details.



User Name	The user name that this user should log in with, or use to run REST API calls
Email	The user's email
Admin	When set, this user will have admin privileges and therefore have access to a wider range of features and REST API calls
Password	The user's password. The user will need this to log in or run REST API calls.

Configuring a Mail Server

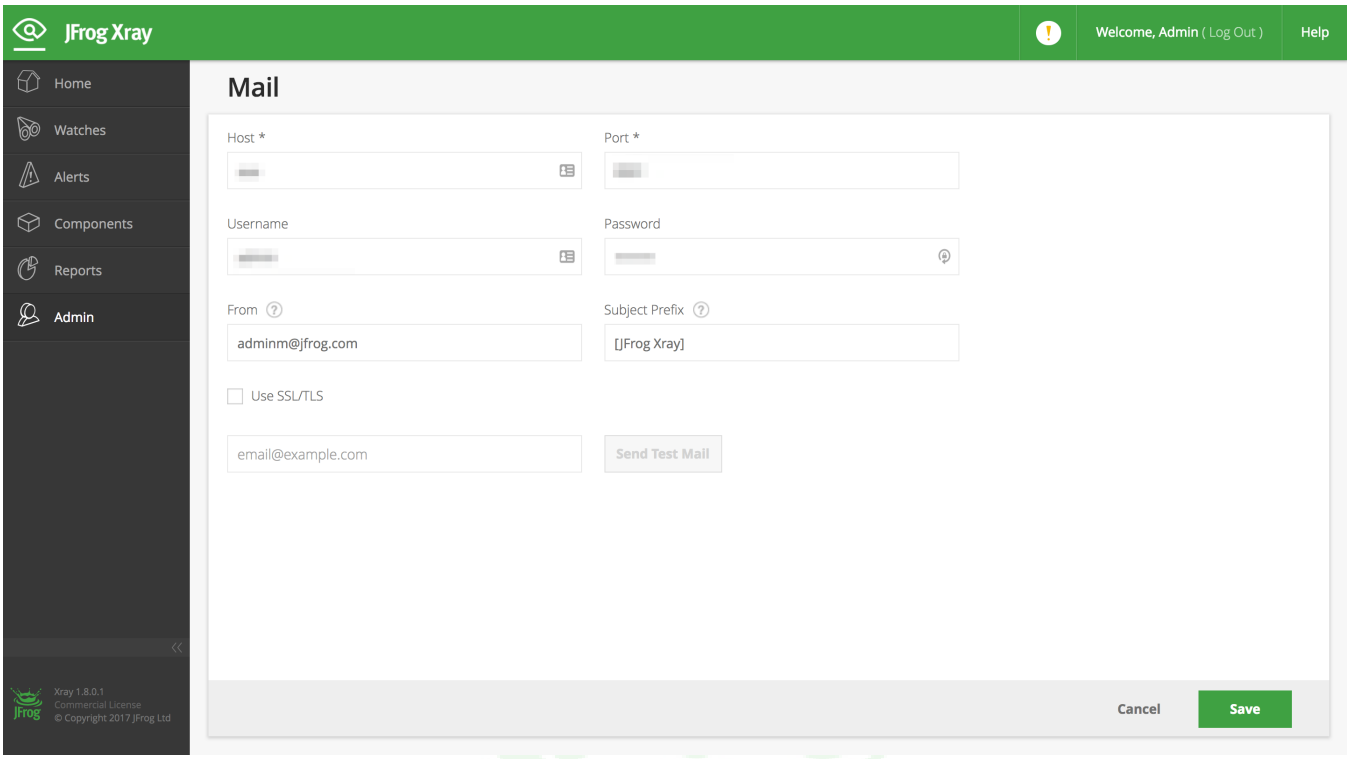
Xray sends email notifications to users for different events that occur.

> **SaaS users**
>
> This configuration is not available for SaaS users where a default mail server is automatically configured.

Some examples are:

- Watchers can send alerts by email.
- Users may be notified about changes in their account information

To enable mail notifications, you need to configure Xray with your mail server details under **Admin I Mail** as described below.

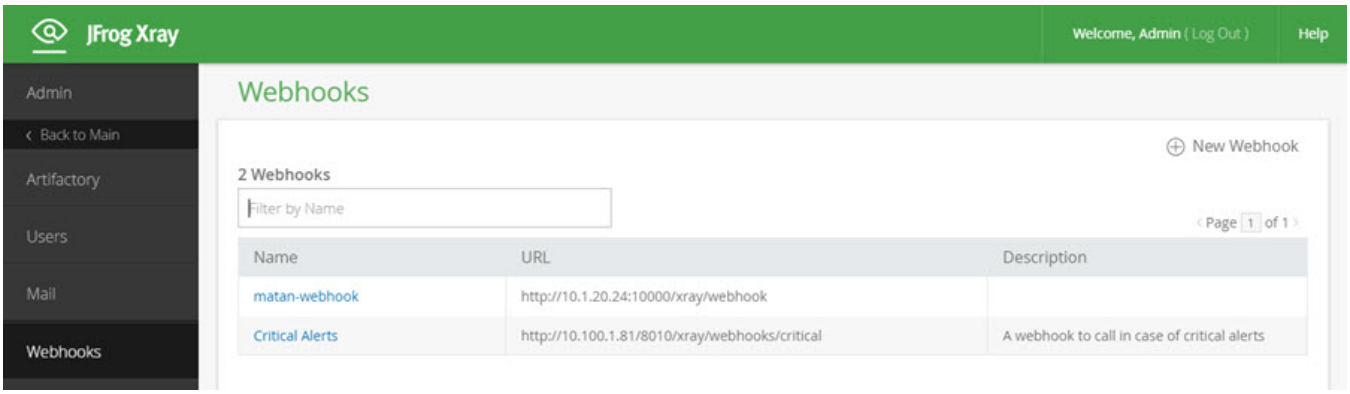


Host	The host name of the mail server.
Port	The port of the mail server.
Username	The username for authentication with the mail server.
Password	The password for authentication with the mail server.
From (optional)	The "from" address header to use in all outgoing mails.
Subject Prefix	A prefix to use for the subject of all outgoing mails.
Artifactory URL (optional)	The Artifactory URL to use in all outgoing mails to denote links to Artifactory.
Use SSL/TLS	When set, uses Transport Layer Security when connecting to the mail server. If not using SSL, this should remain unset and you also need to set `mailNoSsl:true` in the *`xray_config.yaml`* configuration file.
Send Test Mail	Click to send a test message to the specified email address.

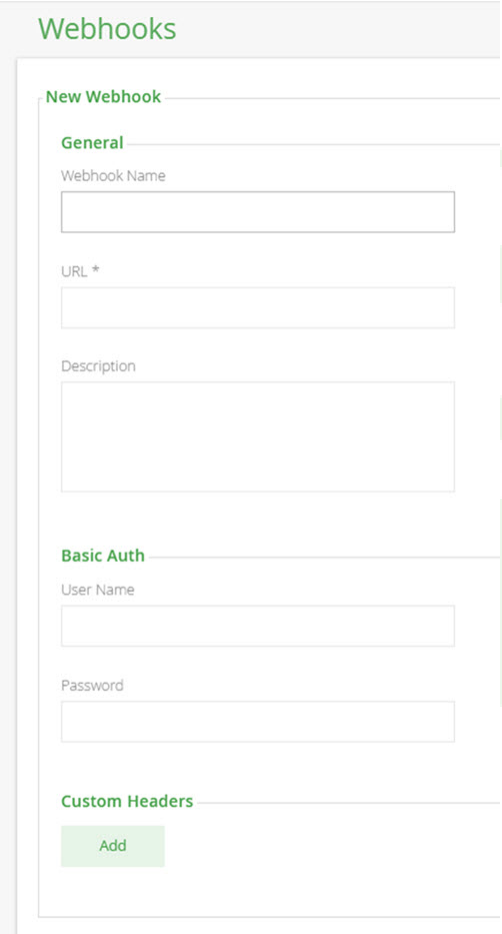
Configuring Webhooks

One of the options when configuring Watchers is to have them invoke webhooks which are proprietary URLs you can define to perform custom actions as a result of an alert being issued.

Webhooks are displayed in the **Admin** module under **Webhooks**.



Click on a webhook to edit its details, or on **New Webhook** to define a new one.



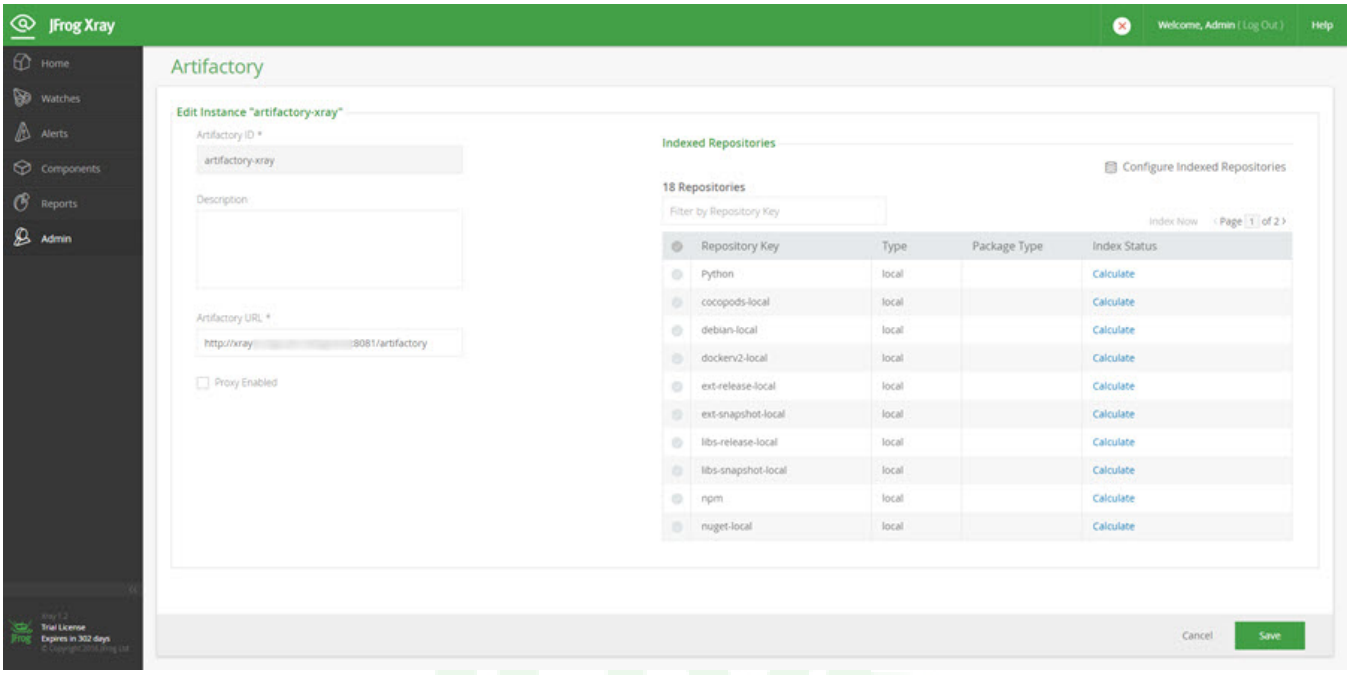
General	
Webhook Name	An identifier for the webhook. This is the name that will be used by any Watches that want to invoke the webhook in case of an alert
URL	The URL that this webhook invokes. The URL is invoked using a payload that describes the alert that was triggered. For a detailed specification of the payload, please refer to Get User Alerts in the Xray REST API.
Description	A free text description
Basic Auth	

User Name /Password	A username and password as required by the webhook
Custom Headers	Any custom headers that may need to be added to invoke the webhook

Indexing Artifacts

JFrog Xray provides information on issues and vulnerabilities in components it has indexed. During normal operation, the database of components is continuously updated and reindexed when changes are made to components in repositories marked for analysis, however, to set up the initial database you need to manually invoke indexing for the repositories that were marked for analysis.

In the **Admin** module, under **Artifactory**, you can view the list of connected Artifactory instances. Click the instance whose repositories you want to index. The details page for that instance shows the list of repositories marked for indexing.



This **Index Status** column will indicate for which repositories indexing is up-to-date, and which need to be indexed. You may click **Calculate** to initiate indexing of a single repository, or check several repositories and click **Index Now** to initiate indexing for multiple repositories at a time.

> **Resource intensive**
>
> Depending on the size of the repository, indexing be a resource intensive operation, so we recommend invoking indexing on repositories separately to avoid excessive loads on your system.

Synchronizing the Database

> **Using a firewall?**
>
> If you are using a firewall, to allow the database sync to complete successfully, you need to add the following URLs to your firewall's whitelist:
>
> 1. https://dl.bintray.com
> 2. https://akamai.bintray.com
> 3. https://jxray.jfrog.io
>
> To test the ability to sync, run the following REST API endpoint:
>
> ```
> https://jxray.jfrog.io/api/v1/system/ping
> ```

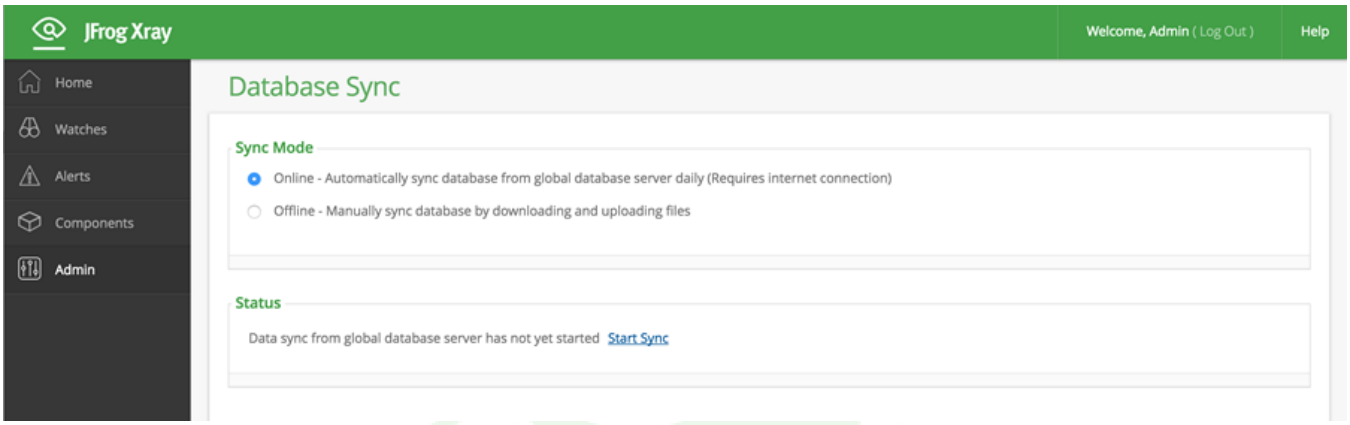
For Xray to scan your indexed artifacts it must ingest data on issues and vulnerabilities from the various feeds it is connected to. The primary feed comes from the **global database server** maintained by JFrog (additional feeds include direct connections you may have through Xray's integrations wi th external providers). There are two ways to synchronize Xray with the global database server:

- **Online:** In online mode, Xray synchronizes with the global database server automatically on a daily basis through an internet connection
- **Offline:** In offline mode, you manually download files from the global database server and then upload them to Xray

To configure synchronization with the global database server, in the **Admin** module, select **Database Sync**.

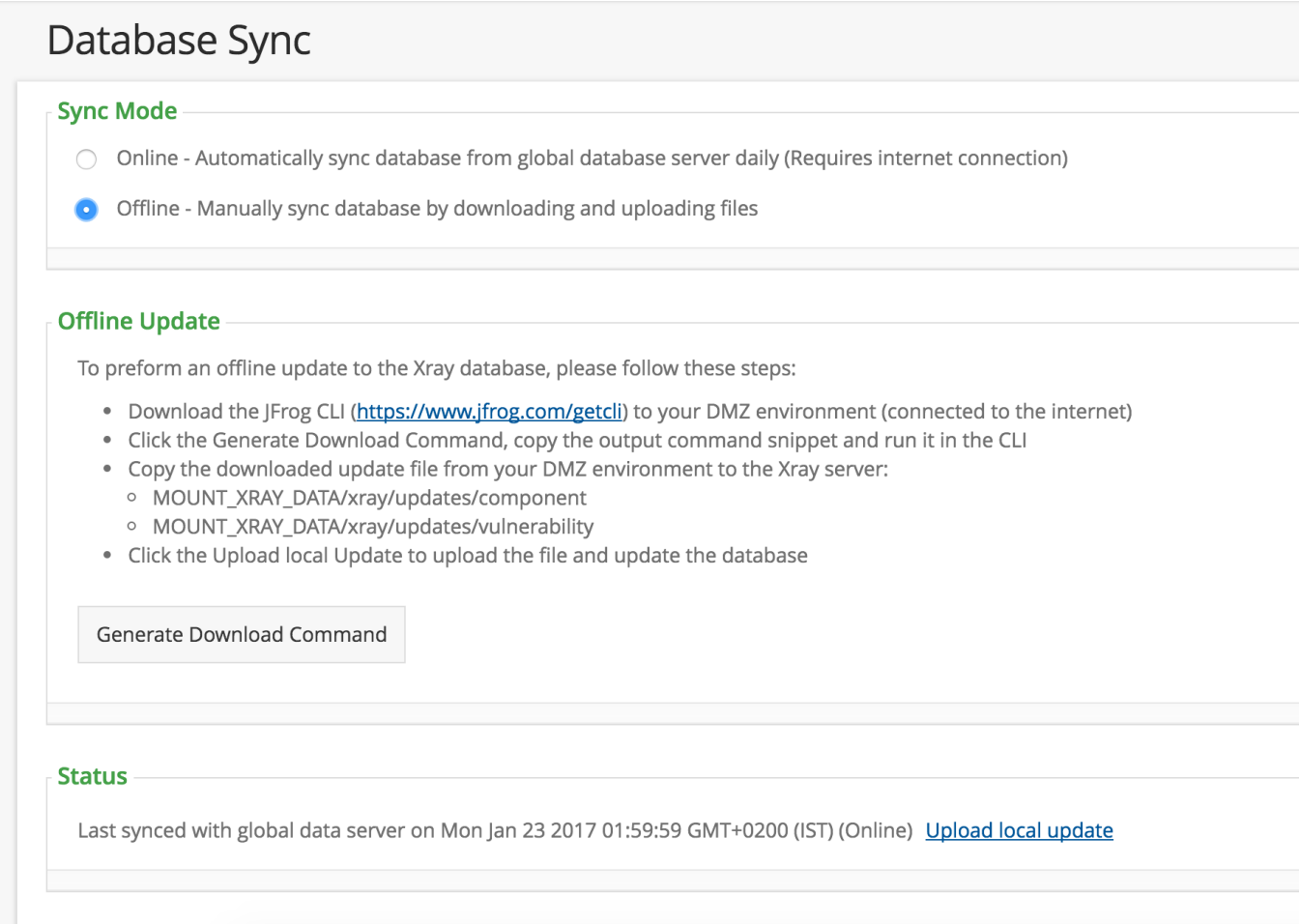
Online Synchronization

To get started right away so Xray can scan your artifacts, you may invoke the initial synchronization manually by selecting **Start Sync** in the **Status** pa nel. From then on, Xray will synchronize issues and vulnerabilities regularly and automatically, once a day.



Offline Synchronization

If, for any reason, you do not want to maintain a live internet connection to the global database server, select **Offline** in the **Sync Mode** panel to get detailed instructions on how to get the latest data available.



Pausing and Resuming Synchronization

Updating Xray with the latest data from the global database server may be a resource intensive operation. During an update, you may click the corresponding link in the **Status** panel to pause the operation at any time to free up resources, and then resume it later.

Working with SSL Using Self-Signed Certificates

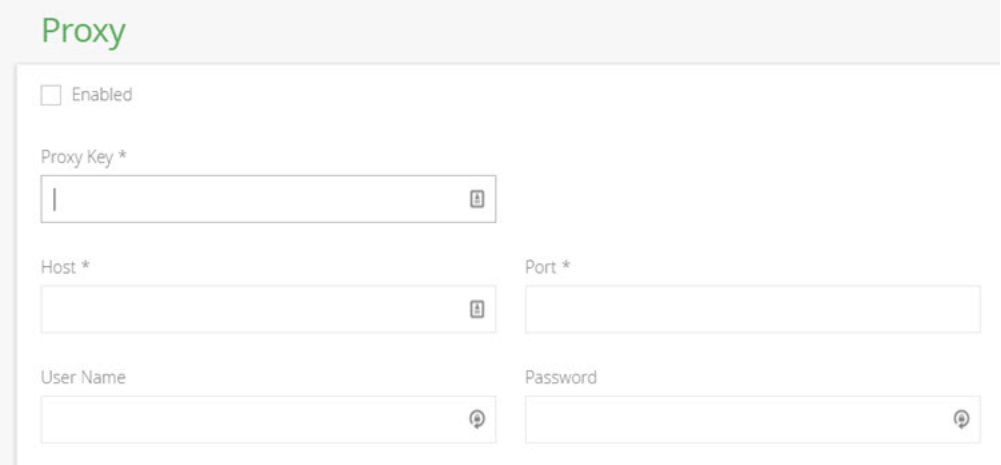
Xray is able to work over a secure connection when connecting to other applications and services. For example, it connects to Artifactory through HTTPS. As part of the HTTPS protocol, Xray is able to verify the site's identity by validating its SSL certificate, however, in cases of a trusted connection, a site may use a self-signed certificate which cannot be validated (which may be the case in an Artifactory/Xray connection).

To skip having to validating a site's self-signed SSL certificate, you may add the `sslInsecure` parameter in Xray's *`xray_config.yaml`* configuration file, and set it to "true" (default, false) as follows:

```
---
    ver:                            1.0
    XrayServerPort:                8000
    mqBaseUrl:                     amqp://guest:guest@rabbitmq:5672/
    mongoUrl:                      mongodb://xray:password@mongodb:27017/?authSource=xray&authMechanism=SCRAM-
SHA-1
    webFolder:                     /opt/jfrog/xray
    postgresqlUrl:                 postgres://xray:xray@postgres:5432/xraydb?sslmode=disable`
        sslInsecure:                        true
```

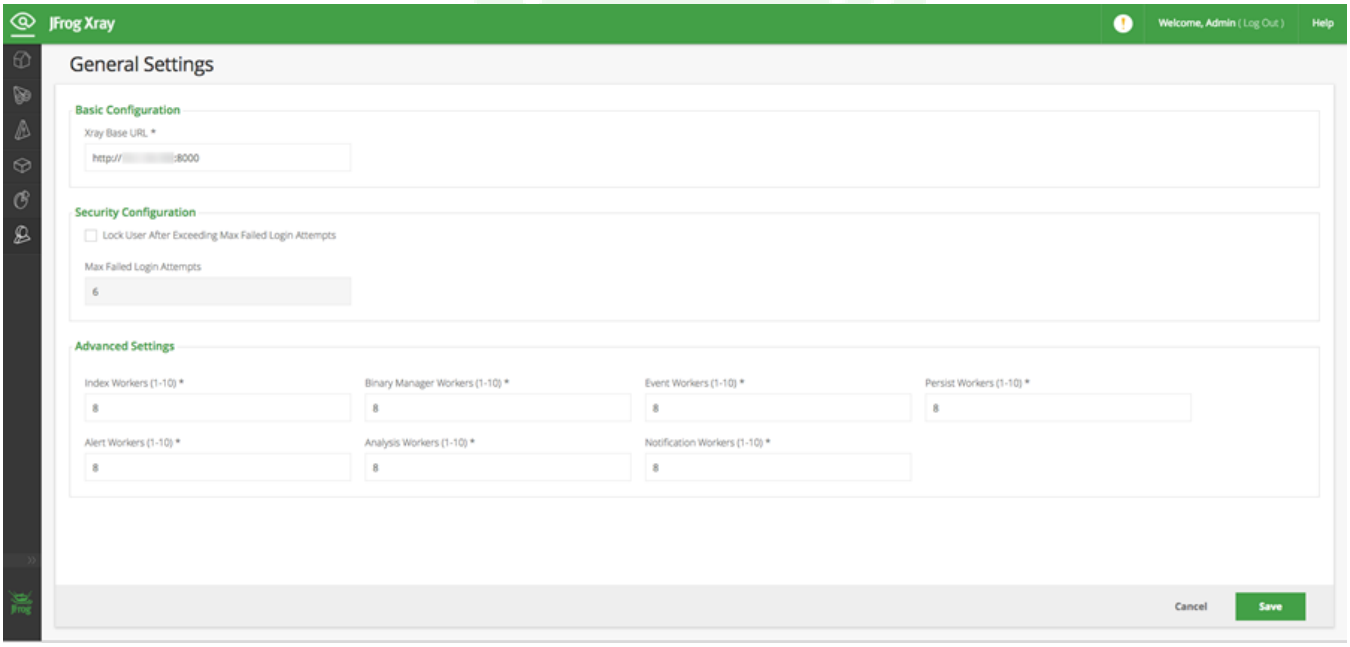
Configuring a Proxy

Depending on your organization's policies, you may need to configure Xray to access external resources through a proxy which may be configured in the **Admin** module under **Proxy.**



General Settings

Xray is built on a set of microservices that perform its actions in the realm of indexing artifacts, communicating with Artifactory, managing events and notifications and so on.



Basic Configuration

Xray Base URL	The base URL through which Xray can be accessed. The base URL uses the following format: PROTOCOL://IP_OR_HOST:8000 For example, `http://10.120.12.123:8000` or `http://XRAY_HOST:8000` Note that the Xray Base URL should not be defined as "localhost", or "127.0.0.1", and should not include the "/web" element. For example, the following base URLs **would NOT work**: `http://127.0.0.1:8000`, `http://10.120.12.123:8000/web/#/home` **Xray access URL is not its base URL** Be careful not to confuse Xray's access URL with its base URL. Xray's access URL is: <XRAY_BASE_URL>/web/#/home If you set the access URL in the Xray Base URL field, connected Artifactory instances will not be able to communicate with Xray

Security Configuration

Lock User After Exceeding Max Failed Login Attempts	Enables locking out users who repeatedly fail logging in according to Max Failed Login Attempts.
Max Failed Login Attempts	The maximum number of login attempts that a user may fail before being locked out of their account when account locking is enabled.

Advanced Settings

Advanced Settings offers several parameters you may configure to tweak the performance of Xray by changing the number of workers performing the different tasks.

Index Workers	The number of workers managing indexing of artifacts.
Binary Manager Workers	The number of workers managing communication with Artifactory.
Event Workers	The number of workers handling events issued by Artifactory to Xray.
Persist Workers	The number of workers managing persistent storage needed to build the artifact relationship graph.
Alert Workers	The number of workers managing alerts.
Analysis Workers	The number of workers involved in impact analysis to determine how a component with a reported issue impacts others in the system.
Notification Workers	The number of workers managing notifications.

For more details on how adjusting these parameters may affect your system's performance, please contact JFrog Support.

Changing Third Party Service Credentials

Xray works with a number of third party services, such as various databases, which come pre-configured with default credentials for access by Xray. The following sections show how to change these default credentials.

MongoDB

To change the default credentials used by Xray to access its internal MongoDB database, you need to log into the database as the "xray" user and change the password as follows:

```
# Access MongoDB as the Xray user
$ mongo --port 27017 -u "xray" -p "password" --authenticationDatabase "xray"

# Switch to the xray database
$ use xray

# Update the credentials
$ db.updateUser("xray",{pwd: "<new_password>"})

# Verify the update was successful by logging in with the new credentials
$ mongo --port 27017 -u "xray" -p "<new_password>" --authenticationDatabase "xray"
```

PostgreSQL

To change the default credentials used by Xray to access its internal PostgreSQL database, you need to log into the database as the "xray" user and change the password as follows:

```
# Access PostgreSQL as the Xray user adding the optional -W flag to invoke the password prompt
$ psql -d xraydb -U xray -W

# Securely change the password for user "xray". Enter and then retype the password at the prompt.
\password xray

# Verify the update was successful by logging in with the new credentials
$ psql -d xraydb -U xray -W
```

RabbitMQ

Xray comes pre-installed with RabbitMQ using its default credentials of: **User:** guest, **Password:** guest

To change the default credentials, follow the steps below:

```
# Change the default password
$ rabbitmqctl change_password guest <new_password>

# Verify the update was successful
$ rabbitmqctl authenticate_user guest <new_password>
```

Watch the Screencast

Watch this screencast and learn how to troubleshoot your Xray installation for problems related to proxy connections, database, network and compute resources.

Authentication

Overview

From **Xray version 1.9 and Artifactory version 5.6**, Xray lets you manage user authentication through one of the Artifactory instances it is connected to. This opens up the ability to use the LDAP/Crowd or SAML corporate authentication facilities that Artifactory uses which, in turn, lets you import users and groups defined in the corresponding LDAP/Crowd or SAML authentication server. The Artifactory instance through which you authenticate users is known in Xray as your "Authentication Provider".

This is in addition to the ability to define users in Xray and provide them with login credentials.

Method of Authentication

The method Xray uses to authenticate a user trying to log in depends on whether Xray is configured with an Authentication provider, and if so, the authentication mechanisms configured (LDAP/Crowd or SAML) for that authentication provider as described below.

Order of Attempting Authentication

In any case, Xray will always try authenticate a user with the following order:

1. **SAML**
   If a SAML server is configured for the Artifactory instance set as the authentication provider, the login screen displays an icon letting the user login using SAML. If the user clicks this icon, SAML is used for authentication
2. **Credentials**
   If the user logs in using their username and password then Xray will try to authenticate in the following order
   a. Through LDAP/Crowd if configured for the Artifactory instance set as the authentication provider
   b. Through users defined in the Artifactory instance set as the authentication provider
   c. Through users defined internally in Xray

Page Contents

Without an Authentication Provider

If no authentication provider is configured in Xray, only users defined in Xray can log in according to the credentials defined for them within Xray as described under Managing Users. Clearly, in this case, you can also only specify **Permissions** for internal Xray users.

With an Authentication Provider

If Xray is configured with an authentication provider then it will try to authenticate a user according to the authentication mechanism configured for the corresponding Artifactory instance. In this case, you can also specify **Permissions** for users defined in the LDAP/Crowd or SAML server of the Artifactory instance set as the authentication provider, as well as for the authentication provider's internal users.

Using SAML

If the authentication provider uses SAML SSO, then the Xray login screen will display a button that the user can use to log in and will be authenticated against the SAML server configured in the corresponding Artifactory instance.

SAML is not compulsory

Even if Xray is configured with an authentication provider that uses SAML, a user can try to login by entering their username and password. In this case, Xray will authenticate the credentials as described in **Order of Attempting Authentication**.

Using LDAP/Crowd

If the authentication provider uses LDAP/Crowd, then the user will enter login credentials using the Xray login screen, but will be authenticated through the LDAP/Crowd server configured in the corresponding Artifactory instance. If authentication fails, Xray will then try to authenticate the credentials provided as described in **Order of Attempting Authentication**.

Using LDAP/Crowd and SAML Together

While it is not a typical scenario, an Artifactory instance that has been set as the authentication provider for Xray may be configured with both an LDAP /Crowd and a SAML server. In this case, the Xray login screen will display a button that the user can use to log in and will be authenticated against the SAML server.

If the user prefers to enter login credentials, Xray will try authenticate them through the LDAP/Crowd server configured in the Artifactory instance set as the authentication provider. If authentication fails, Xray will then try to authenticate the credentials as described in **Order of Attempting Authentication**.

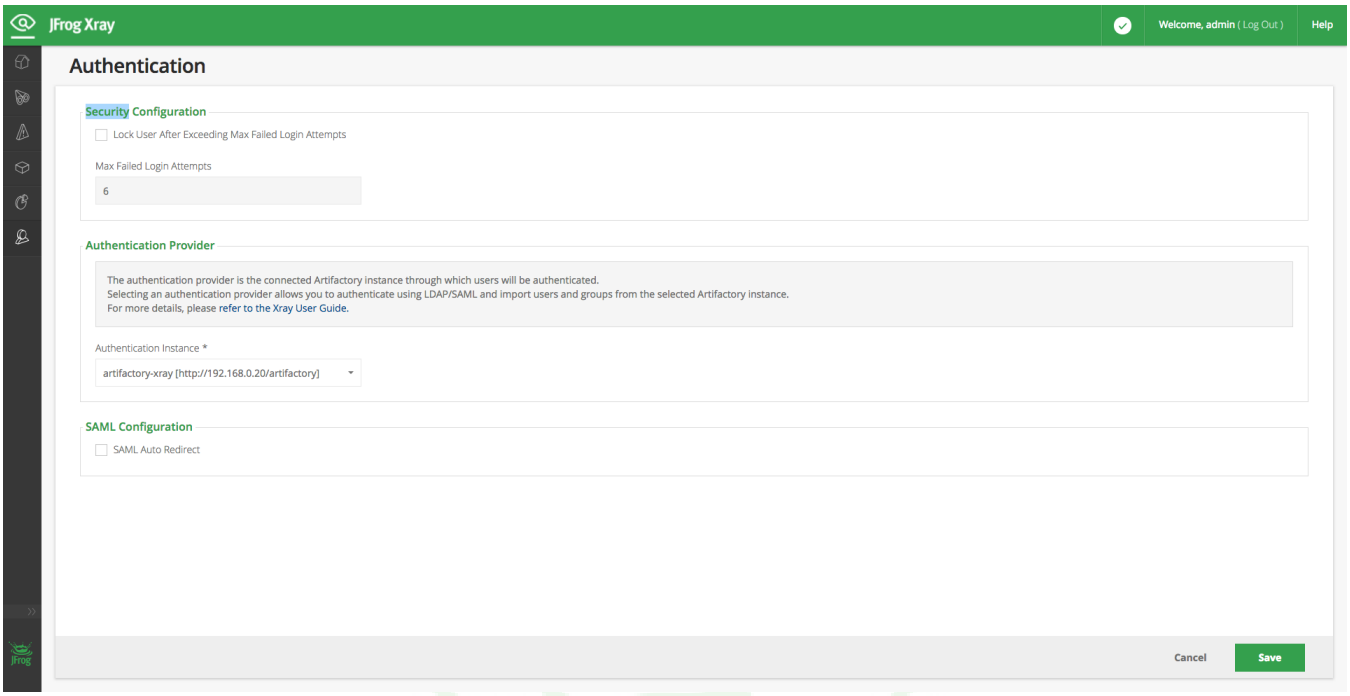
Configuring an Authentication Provider

You can set any of the Artifactory instances to which Xray is connected as the Authentication Provider.

Selecting an Authentication Provider

To set an authentication provider, in the **Admin** module, select **Security I Authentication.**

In the **Authentication** screen**,** under **Authentication Provider,** the **Authentication Instance** field displays the connected Artifactory instances. Select one of those instances to be the authentication provider.



Once the authentication provider is set, Xray will authenticate users logging in as described above.

SAML Auto Redirect

When set, Xray will try to log the user in through SAML. If the user is already logged in through SAML (through the connected Artifactory instance, or any other application), Xray will automatically log them in using the same SAML server for authentication. If the user is not logged in, Xray will display the SAML login screen.

Permissions

Overview

From version 1.9, JFrog Xray offers a flexible permissions model that gives an administrator fine-grained control over how users and groups access the different features of Xray.

> **Authentication Provider Recommended**
>
> While it is not compulsory to specify an Authentication Provider, it is a recommended best practice. Through an authentication provider, you can apply permissions to all users defined in your LDAP or SAML servers, and also those internally defined in the corresponding Artifactory instance. Without an authentication provider, you can only apply permissions to users internally defined in Xray.

Permissions are managed as a set of rules applied to three vectors: Resources, Users/Groups and Actions.

Resources

Resources define the scope of a permission and specify the repositories and builds in the connected Artifactory instance to which the permission applies. Xray also lets you specify a **Global Scope** for a permission, and in this case, it applies to all repositories and builds in the connected Artifactory instance. For example, if a rule provides a user with "View Components" permission (see Actions below) on a global scope, it means that user will be able to see the components of all repositories and builds in the system.

Users and Groups

Once the scope of a permission (specific repositories/builds or Global Scope) is specified, you can specify the users and groups to which the permission applies. If you have selected one of the connected Artifactory instances as an **Authentication Provider**, Xray will work with the users and groups defined in the corresponding Artifactory instance. If you are not using an **Authentication Provider**, Xray will work with its own, internally defined users.

Actions

Once you have defined the resources and users/groups to which a permission applies, you can specify the actions that those users/groups can perform on the specified resources. The table below describes the actions you can specify for a permission.

Action	Description
View Components	Allows the specified users/groups to view components on the resources specified in the rule. This applies to any activity related to components such as component search, component details, impact of issues etc. For example, if a repository called "maven-special" is not included in the scope of a permission, users/groups specified in that permission will not see any of the components hosted in that repository. Those components won't turn up in search queries, they won't be displayed in issue analysis etc. Note that this permission is **version-agnostic** which means that users/groups specified in the permission can see all versions of a component, even if some of those versions are in resources outside of the scope defined in the permission.
Manage Components	Allows the specified users/groups to perform actions on components in the specified resources. Currently, the only action available is to manually trigger a scan.
View Watches /Issues	Allows the specified users/groups to see Watches and Issues related to the resources specified in the permission.
Manage Watches /Issues	Allows the specified users/groups to add, edit and delete Watches, and to Ignore Alerts and Issues related to the resources specified in the permission.
Generate Reports	This action can only be applied to a Global Scope. It allows the specified users/groups to view global security and license reports.

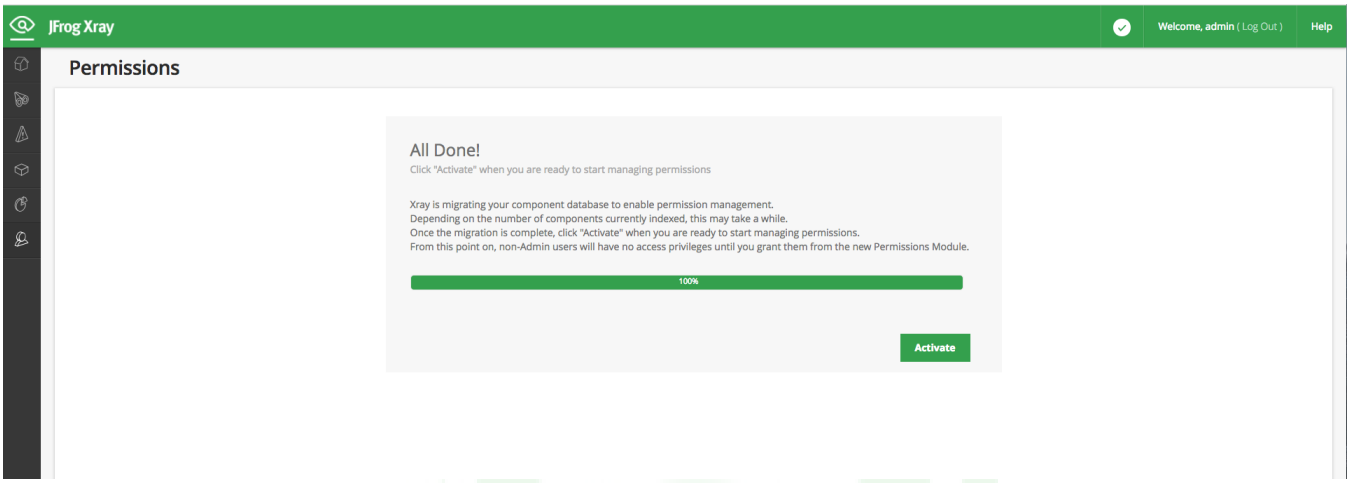
Page Contents

Admin	This action can only be applied to a Global Scope. It allows to view the Admin module and perform all actions available to an Xray administrator such as managing connected Artifactory instances, doing a DB sync etc.

Activating Permission Management

For a clean installation of JFrog Xray version 1.9 and above, permission management is automatically enabled and you can create and edit permissions as described in the sections below.

When upgrading Xray from a version that is below 1.9 to version 1.9 and above, when you start up Xray, it will migrate your component database to enable permission management. This process is initiated automatically by Xray upon startup and may take a while depending on the size of your database, however, the process runs in the background allowing you to continue using the other features of Xray in the mean time. You can view the progress of the migration process in the **Admin** module under **Security | Permissions**.



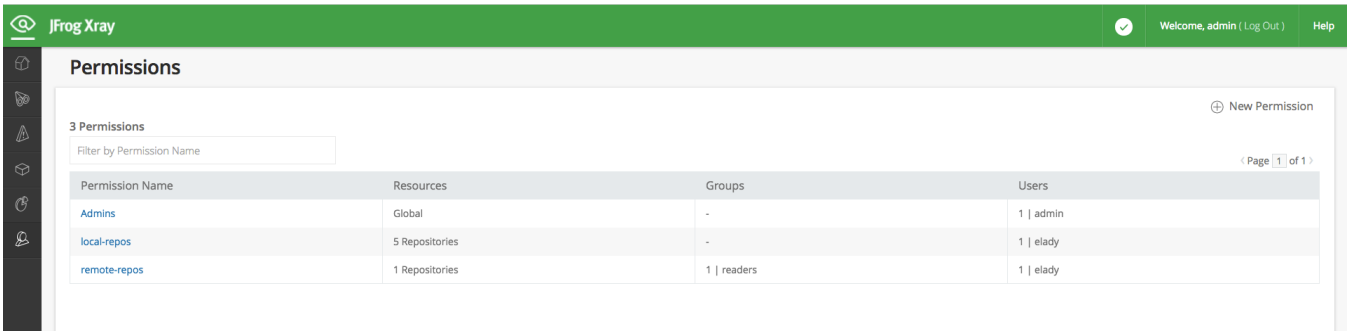
Permission management must be activated to be functional

Once the component database migration is complete, you must activate permission management for it to be functional. Note, however that activating permission management is optional. You may continue using Xray, as before, without any permission management. In this case all users accessing the system will have the same Admin privileges.

Once you activate permission management, you can create and edit permissions as described in the sections below.

Creating and Editing Permissions

You can access the list of Permissions defined in Xray from the **Admin** module under **Security | Permissions.**



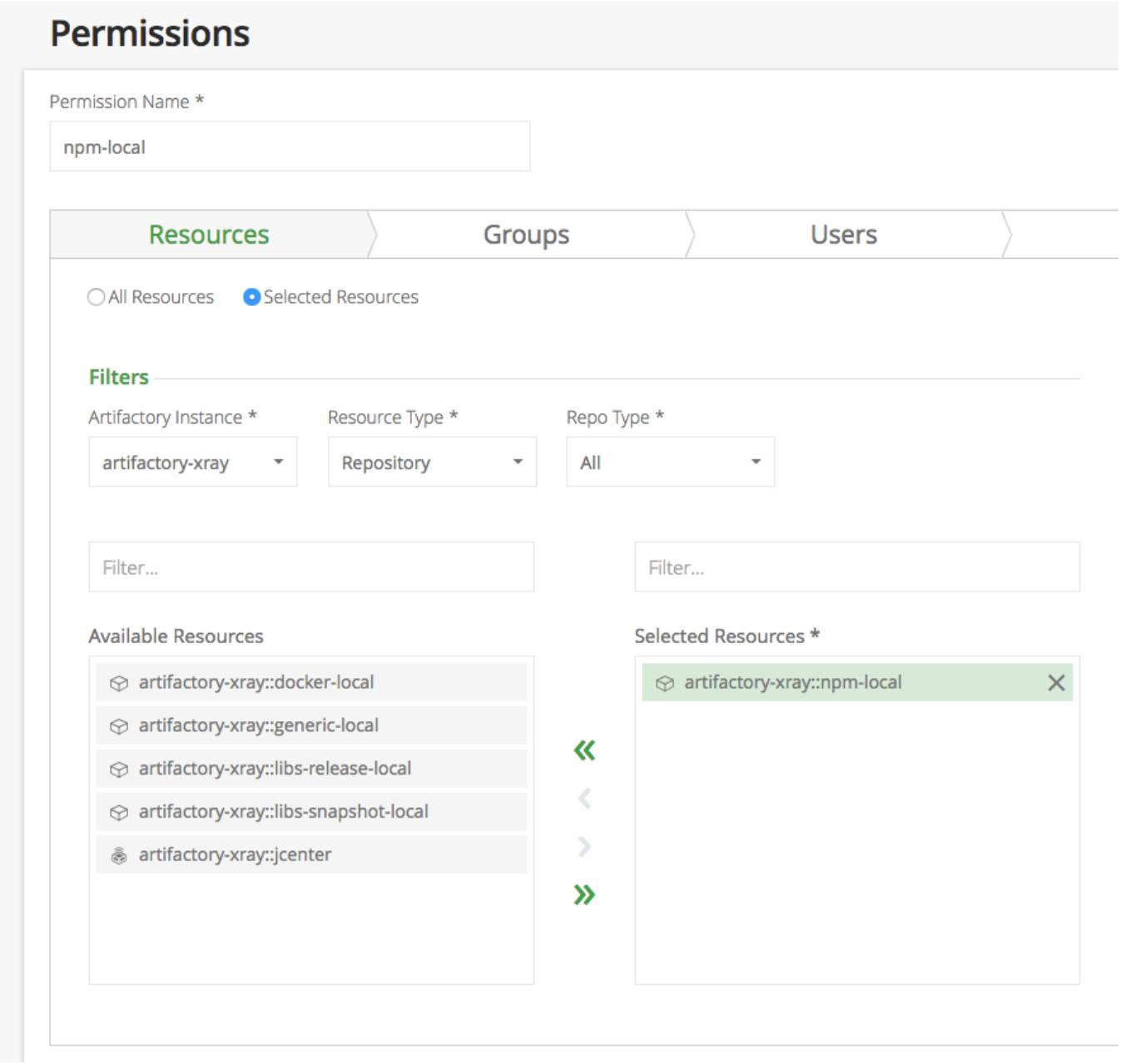
Double-click a Permission Name to edit an existing Permission, or click "New Permission" to create a new one.

Creating editing a permission is done in three steps.

1. Specifying Resources
2. Specifying Groups and Actions
3. Specifying Users and Actions

After completing these steps, make sure to click "Save & Finish" to save your changes.

Specifying Resources



Permission Name	A logical name for this permission.

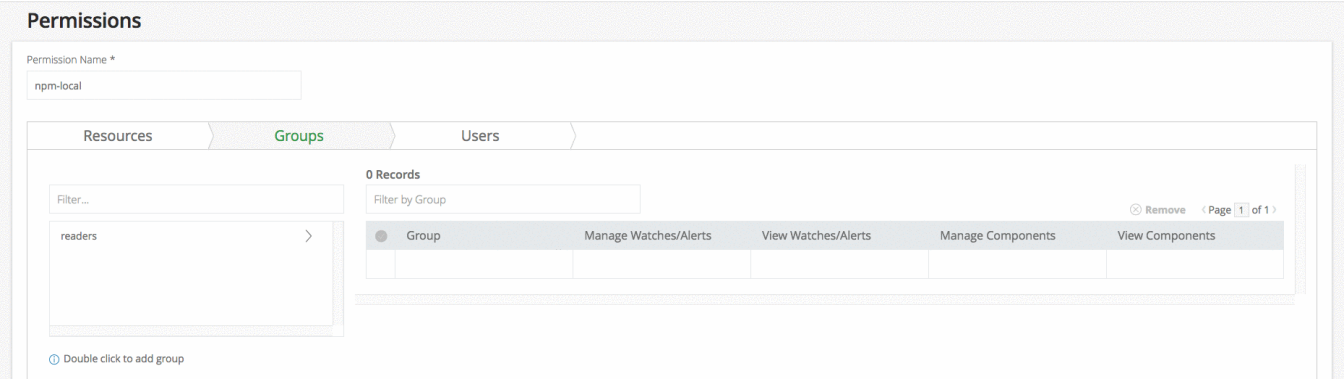
All Resources	If selected, this permission applies to all resources available. When selected, the rest of this form is disabled since there is nothing more to specify.
Selected Resources	If selected, you need to specify the resources (Artifactory instances, repositories and/or builds) to which this permission applies.
Filters	Gives you control over which resources this permission should apply. Filters Artifactory Instance * All Resource Type * All Filter... Filter... Available Resources Selected Resources *
Available Resources	Displays the resources available for this permission according to the filters you have applied.
Selected Resources	Displays the resources you have selected for this permission.

Once you have specified the resources for this permissions, select the **Groups** tab to specify the groups on which to apply it.

Specifying Groups and Actions

The **Groups** tab will display groups defined in the Artifactory instance specified as your authentication provider.

Using the arrow, or by double-clicking, add the Groups for which you want to define actions and then specify the actions allowed.



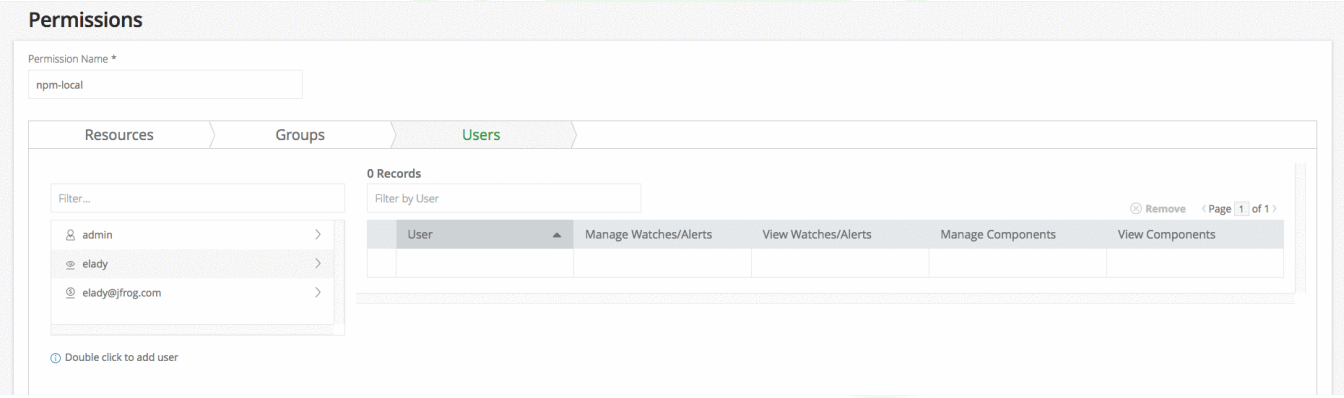
Once you have specified Groups and their allowed actions for this permission, select the **Users** tab to specify additional users on which to apply it.

Specifying Users and Actions

The **Users** tab will display uses defined in the Artifactory instance specified as your authentication provider as well as any other users defined internally in Xray.

Note that the list of users indicates where each user is defined. In the example below, we can see that the user called elady@jfrog.com is imported from the connected Artifactory instance defined as the Authentication Provider which is using SAML for authentication.

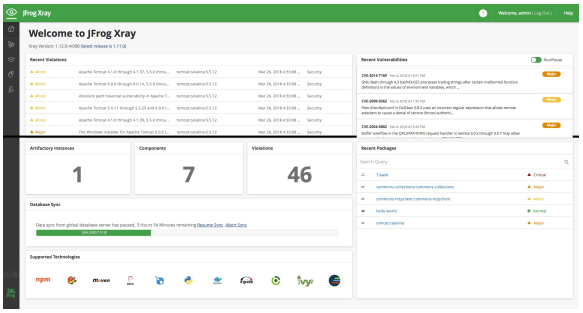
Using the arrow, or by double-clicking, add the users for which you want to define actions and then specify the actions allowed.



Home

Overview

Xray displays the **Home** screen when you log in. The Home screen is a dashboard that gives you an instant picture showing a number of critical parameters in your Xray setup



Page Contents

Top Ribbon

The top ribbon in Xray displays the logged-in user, and provides access to Help. Admin users may click the status icon to get a quick and easy view of general system status.

Recent Violations

The **Recent Violations** panel displays a list of the most recent Violations that were issued by Xray according the watches that the currently logged-in user has defined.

Recent Vulnerabilities

The **Recent Vulnerabilities** panel displays the most recent new vulnerabilities that have been added to Xray, whether to the Global Database Server, or through one of the external integrations.

Basic Information

Basic information is displayed in three panels that show:

- the number of Artifactory instances connected to this instance of Xray
- the total number of Artifacts currently indexed by Xray (for all of the connected Artifactory instances),
- the total number of Violations that are currently active in the system

Database Sync Status

The **Database Sync** panel displays the current status of synchronization with the Global Database Server, and provides a link that lets you initiate a synchronization process.

Supported Technologies

The **Supported Technologies** panel displays the list of technologies that Xray scans and indexes for vulnerabilities.

Recent Packages

The **Recent Packages** panel displays new packages that have recently entered the system and have been indexed for scanning by Xray. At the top of the panel you can enter a search query to search for a particular package. Xray will take you to the **Components** module where it will display the results of your search. In the **Components** module, you can drill down to view details of any components that were found to match your search query, or run a new search using all the parameters available.

Watches

Overview

Watches monitor artifacts for issues, and trigger alerts if any are found based on two types of analysis which Xray performs

Scanning

Xray monitors builds or repositories in Artifactory for issues. Each time a monitored build is updated, or an artifact is deployed to a monitored repository, Xray will scan its dependencies and trigger a violation if any dependency with issues is found.

Impact Analysis

Xray listens to all providers currently streaming feeds regarding issues. If any provider notifies Xray of a new issue with an artifact, Xray looks up the artifact in its database. If the artifact is already in the database, Xray will perform an impact analysis to determine all the artifacts in Artifactory that are ultimately affected by the issue by virtue of their including the problematic artifact. The results are displayed in an impact analysis graph.

Focusing on Specific Components (Filters)

An active Artifactory instance may cause Xray to trigger many alerts on artifacts which are not interesting to you. To focus on artifacts you want to monitor, you can fine-tune Xray to trigger violations only for artifacts that pass through **Filters** you define based on the following parameters:

- Regex
- Package type
- Mime type
- Build
- Property
- Allowed and Banned Licenses

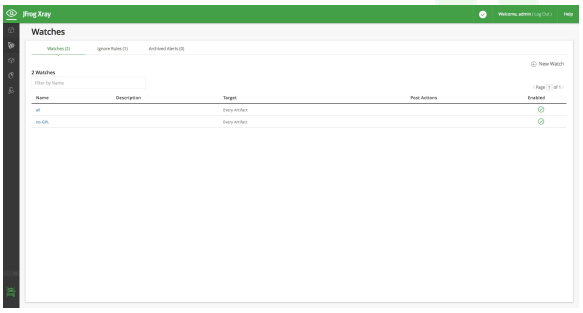
The specific filters available depend on the **Target Type** on which the watch is defined.

In addition, you need to define a Severity Filter to specify the minimum severity for which you want Xray to register a violation.

A watch in Xray will only register a violation for components that:

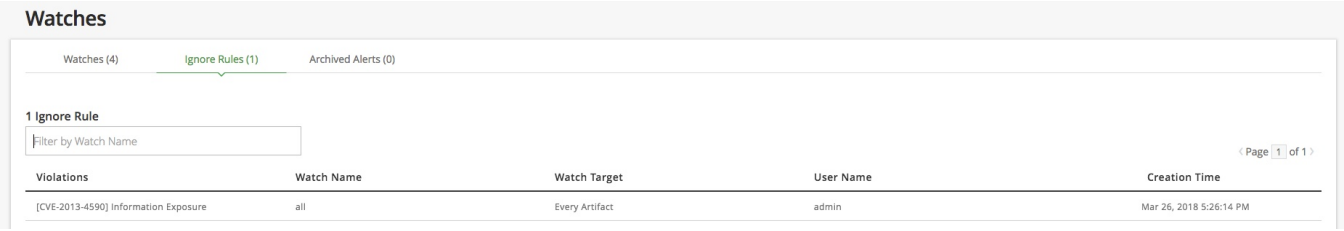
- have been identified by one of Xray's providers to have some kind of vulnerability
- meet all the criteria (i.e. Artifact Filters and/or Severity Filters) you defined for it.

All the Watches defined in your system are displayed in the **Watches** module.



Ignore Rules

The **Ignore Rules** tab displays violations which you have chosen to ignore in the Component Details display.



Archived Alerts

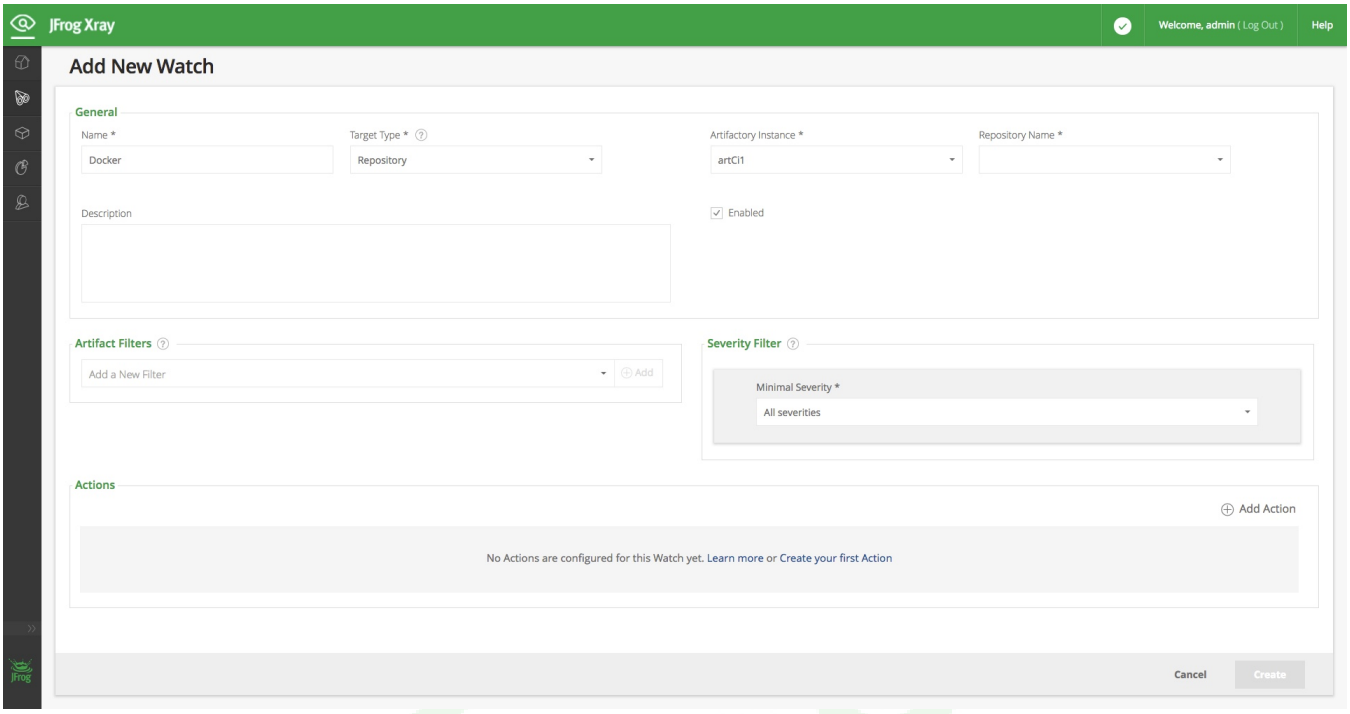
Page Contents

From version 1.12, Xray introduced the concept of Violations which replaces Alerts, and Alerts are not generated any more.

The **Archived Alerts** tab displays all the alerts that Xray generated prior to being upgraded to version 1.12.

Creating and Editing a Watch

To create a new watch, click New Watch and fill in the fields that define the watch.



General	
Name	A logical name for this watch.
Target Type	**Repository:** The watch monitors the repository specified in the **Repository Name** field. **Every Artifact:** The watch monitors all artifacts in all repositories indexed by Xray. **All Builds:** The watch monitors all builds in all Artifactory instances connected to this instance of Xray.
Artifactory Instance	The Artifactory instance to which this watch should be applied. The watch will only take effect if Xray is currently connected to the specified instance.
Repository Name	The build or repository to watch based on the Target Type
Description	A general description of the Watch.
Enabled	When checked, the watch is enabled
Artifact Filters	Specifies which Artifact Filters to apply. Only artifacts matching all filters will trigger a violation.

Severity Filters	Specifies which Severity Filters to apply. Only artifacts detected to have violations that meet the Minimal Severity set or greater will trigger a violation.
Actions	Specifies what additional actions to take once aviolation has been triggered.

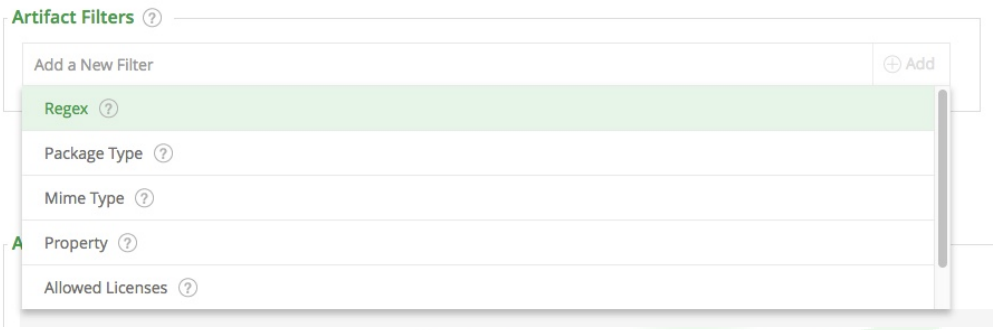
You can edit an existing Watch by clicking its name in the Watches table and editing its parameters in the form displayed.

Artifact Filters

The filters you define for a watch determine which components in the currently observed Artifactory instance will generate alerts and under what conditions. You can define any number of filters, and the watch will only trigger a violation if an artifact meets the condition of all of the filters defined. The following content filters are available:

- Regex: Generate a violation based on a component's name
- Package Type: Generate a violation based on a component's package type
- Mime Type: Generate a violation based on a component's MIME type
- Property: Generate a violation if a component is annotated with the specified property
- Allowed/Banned Licenses: Generate a violation if a component uses a license that is not allowed

To add a filter to your watch, select the filter type and click "Add".



Xray will display the filter for you to specify the parameter to trigger a violation.

> **Pass through ALL filters**
>
> You can define any number of filters for a watch, and only artifacts that pass through all of them will trigger a violation.

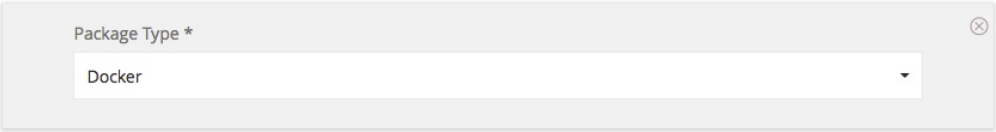
Regex

A **Regex** filter uses a regular expression to specify the name of an artifact. The watch will only trigger a violation if an artifact's name matches the expression.

For example, the filter above specifies that the watch should only trigger a violation for rpm files.

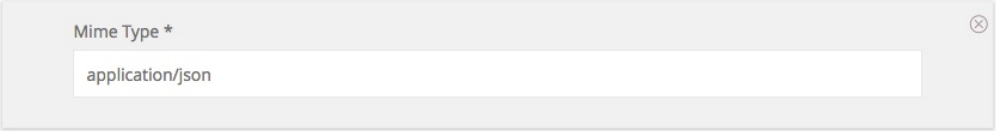
Package Type

A Package Type filter specifies an artifact's package type. The watch will only trigger a violation if an artifact has the specified package type.



Mime Type

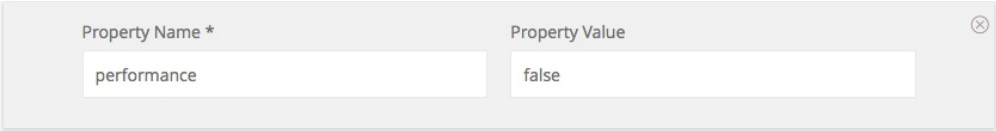
A Mime Type filter specifies an artifact's mime type. The watch will only trigger a violation if an artifact has the specified mime type.



For example, the filter above specifies that the watch should trigger a violation for any artifact with an "application/json" mime type.

Property

A Property filter specifies a property annotating an artifact and its value. The watch will only trigger a violation if the property has the specified value.



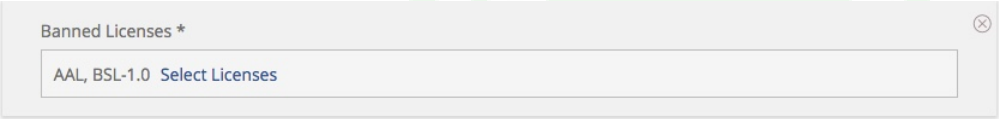
For example, the filter above specifies that the watch should trigger a violation if an artifact with a property named "performance" has the value "false".

Allowed and Banned Licenses

An Allowed Licenses filter specifies a whitelist of OSS licenses that may be attached to an artifact. The watch will only trigger a violation if an artifact has an OSS license other than the ones specified. You may include "Unknown" in the list of allowed licenses to allow components with an unknown license to reside in your repositories without triggering a violation

A Banned Licenses filter specifies a blacklist of OSS licenses that may not be attached to an artifact. The watch will only trigger a violation if an artifact has any of OSS licenses specified. You may include "Unknown" in the list of banned licenses so that components whose license cannot be determined will trigger a violation.

You may specify either an Allowed License filter or a Banned licenses filter for a violation, but not both together. Once you have specified Allowed Licenses or Banned Licenses, use the **Select Licenses** link to specify the licenses to allow or ban.



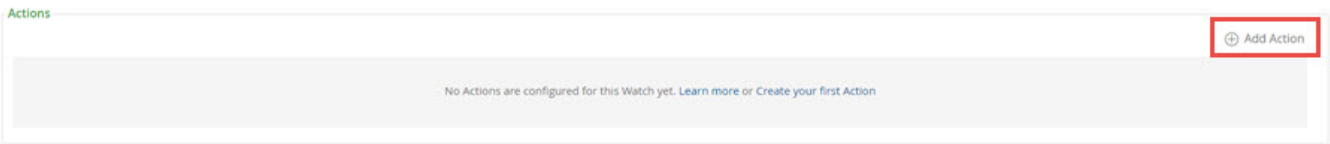
Severity Filter

A Severity filter specifies the minimum severity of an issue associated with an artifact. If an artifact has an issue with an equal or higher severity, a violation is generated.

> **Severity filter is required**
>
> It is compulsory to define a Severity Filter for all watches.

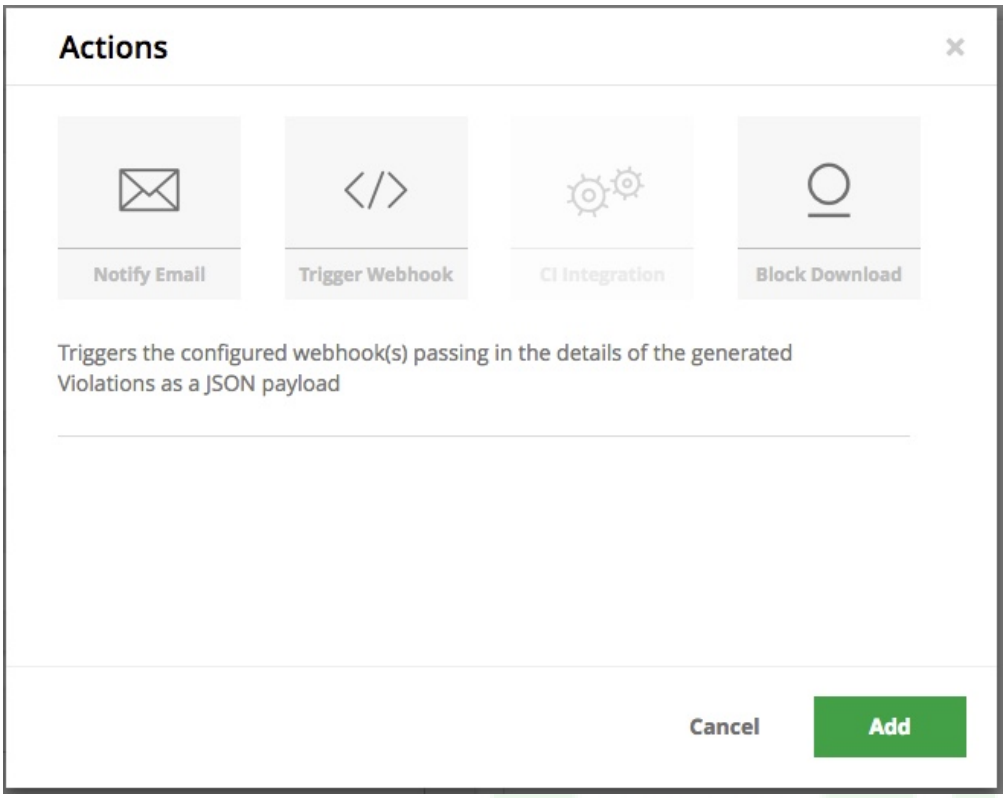
Actions

The Actions panel lets you specify additional actions that Xray should take once a violation has been triggered by watch in which the action is defined. You can specify multiple actions for a violation. To add an action, click **Add Action**.



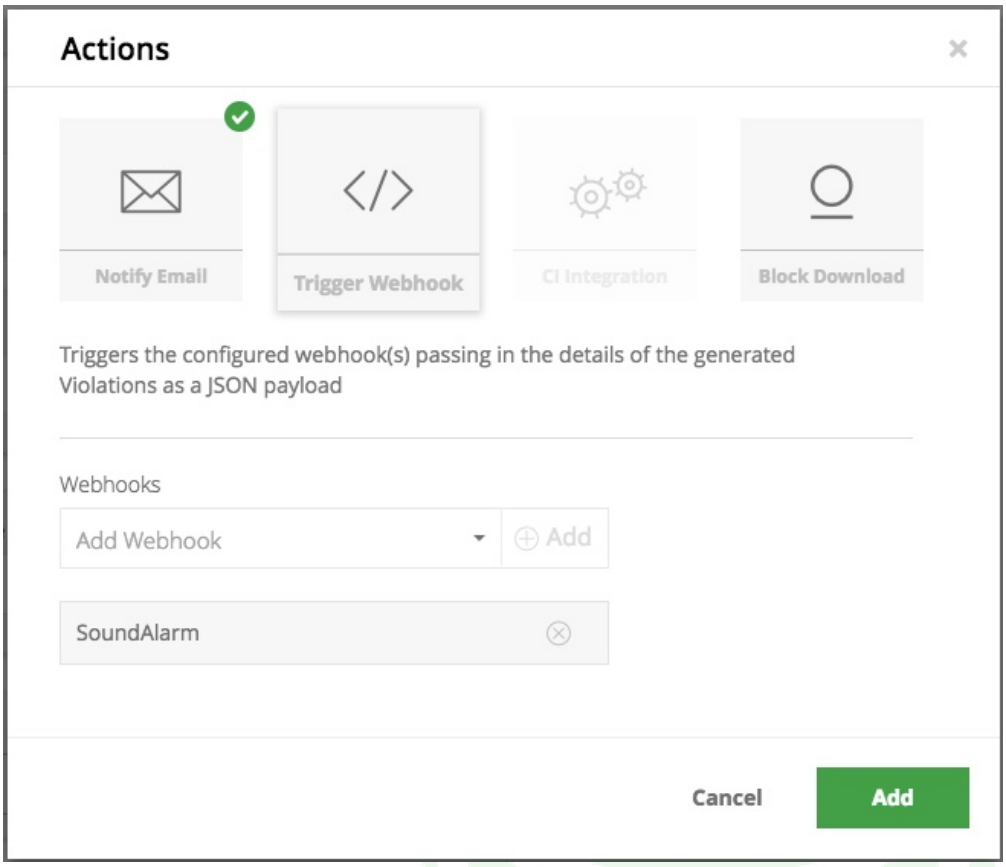
Notify Email

This action lets you specify email addresses to which Xray should send an email message about a violation when one is triggered. For this to work, you need to have a mail server configured in Xray.



Trigger Webhook

This action lets you specify webhooks you have configured in Xray that should be invoked when a violation is triggered.



Webhook Payload

The payload provided to any triggered webhook is a JSON object describing a list of Alerts with the following format:

> **Alerts are being deprecated**
>
> From version 1.12, Alerts are in the process of being deprecated. Currently, the webhook payload still references alerts, however, this will be changed in forthcoming releases

```
{
  "alert_id": "<Alert ID>",
  "created": "<Alert creation time stamp in ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
  "top_severity": "<Top severity of any issue in the alert>",
  "watch_name": "<Logical name for the watch>",
  "issues": [
        {
                "severity": "<Issue severity>",
            "type": "<Issue type>",
            "provider": "<Issue provider>",
            "created": "<Issue creation time stamp in ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
            "summary": "<Issue summary>",
              "description": "<Issue description>",
              "impacted_artifacts": [
                        {
                              "name": "<Artifact name>",
                          "display_name": "<Artifact dispalay name>",
                          "path": "<Artifact path in Artifactory>",
                          "pkg_type": "<Package type>",
                          "sha256": "<Artifact SHA 256 checksum>",
                          "sha1": "<Artifact SHA 1 checksum>",
                          "depth": <Artifact depth in its hierarchy>,
                         "parent_sha": "<Parent artifact SHA 1 checksum>",
                                 "infected_files": [
                            {
                                       "name": "<File name>",
                                       "path": "<File path>",
                                       "sha256": "<File SHA 256 checksum>",
                                       "depth": <File depth in hierarchy>,
                                       "parent_sha": "<File's parent SHA 1 checksum>",
                                       "display_name": "<File's display name>",
                                       "pkg_type": "File's package type"
                            }
                                ]
                        }
                ]
        }
  ]
}
```

The following shows an example payload for a webhook

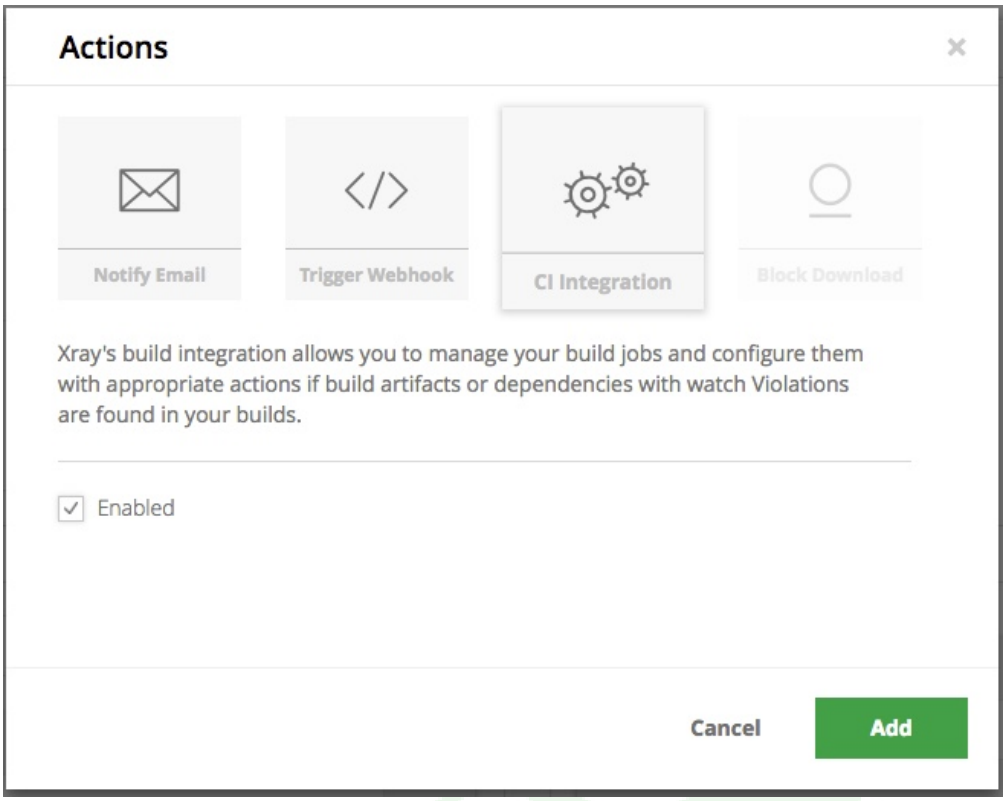
```
{
  "alert_id": "5aa6d687db80740001ac83b4",
  "created": "0001-01-01T00:00:00Z",
  "top_severity": "Critical",
  "watch_name": "no-Apache-2.0-builds",
  "issues": [
    {
      "severity": "Critical",
      "type": "security",
      "provider": "Custom",
      "created": "2018-03-12T19:12:06.702Z",
      "summary": "custom-glassfish",
      "description": "custom-glassfish",
      "impacted_artifacts": [
        {
          "name": "test",
          "display_name": "test:6639",
          "path": "artifactory-xray/builds/",
          "pkg_type": "Build",
          "sha256": "c9be3f74c49d2f3ea273de9c9e172ea99be696d995f31876d43185113bbe91bb",
          "sha1": "737145943754ac99a678d366269dcafc205233ba",
          "depth": 0,
          "parent_sha": "c9be3f74c49d2f3ea273de9c9e172ea99be696d995f31876d43185113bbe91bb",
          "infected_files": [
            {
              "name": "ant-1.9.4.jar",
```

```
              "path": "",
              "sha256": "649ae0730251de07b8913f49286d46bba7b92d47c5f332610aa426c4f02161d8",
              "depth": 0,
              "parent_sha": "c9be3f74c49d2f3ea273de9c9e172ea99be696d995f31876d43185113bbe91bb",
              "display_name": "ant-1.9.4.jar",
              "pkg_type": "Generic"
            },
            {
              "name": "aopalliance-repackaged-2.4.0-b09.jar",
              "path": "",
              "sha256": "a97667a617fa5d427c2e95ce6f3eab5cf2d21d00c69ad2a7524ff6d9a9144f58",
              "depth": 0,
              "parent_sha": "c9be3f74c49d2f3ea273de9c9e172ea99be696d995f31876d43185113bbe91bb",
              "display_name": "org.glassfish.hk2.external:aopalliance-repackaged:2.4.0-b09",
              "pkg_type": "Maven"
            }
          ]
        }
      ]
    },
    {
      "severity": "Critical",
      "type": "License",
      "summary": "Apache-2.0",
      "description": "Apache License 2.0",
      "impacted_artifacts": [
        {
          "name": "test",
          "display_name": "test:6639",
          "path": "artifactory-xray/builds/",
          "pkg_type": "Build",
          "sha256": "c9be3f74c49d2f3ea273de9c9e172ea99be696d995f31876d43185113bbe91bb",
          "sha1": "737145943754ac99a678d366269dcafc205233ba",
          "depth": 0,
          "parent_sha": "c9be3f74c49d2f3ea273de9c9e172ea99be696d995f31876d43185113bbe91bb",
          "infected_files": [
            {
              "name": "ant-1.9.4.jar",
              "path": "",
              "sha256": "649ae0730251de07b8913f49286d46bba7b92d47c5f332610aa426c4f02161d8",
              "depth": 0,
              "parent_sha": "c9be3f74c49d2f3ea273de9c9e172ea99be696d995f31876d43185113bbe91bb",
              "display_name": "ant-1.9.4.jar",
              "pkg_type": "Generic"
            },
            {
              "name": "aopalliance-repackaged-2.4.0-b09.jar",
              "path": "",
              "sha256": "a97667a617fa5d427c2e95ce6f3eab5cf2d21d00c69ad2a7524ff6d9a9144f58",
              "depth": 0,
              "parent_sha": "c9be3f74c49d2f3ea273de9c9e172ea99be696d995f31876d43185113bbe91bb",
              "display_name": "org.glassfish.hk2.external:aopalliance-repackaged:2.4.0-b09",
              "pkg_type": "Maven"
            }
          ]
        }
      ]
    }
  ]
}
```


CI Integration

This action lets you specify that if a CI server requests a build to be scanned, and the Watch triggers a violation, Xray will respond with an indication that the build job should fail.

This action is only available if the Watch is defined with an **All Builds** target type.

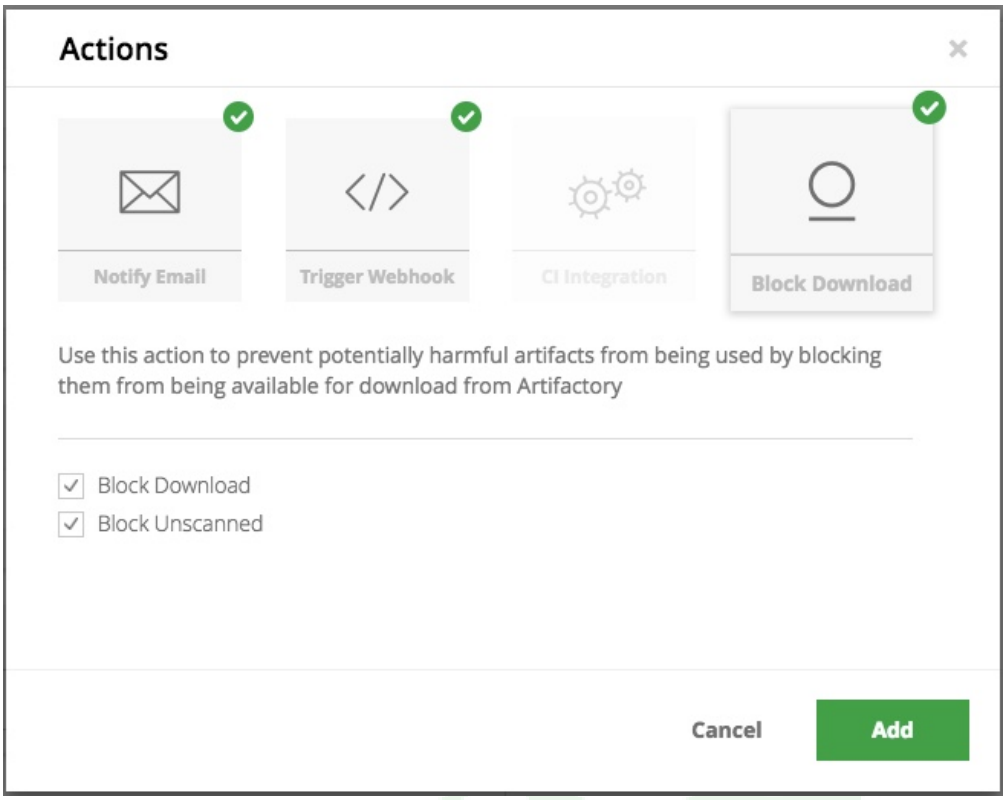


No Fail Build Job Actions defined?

If a request to scan a build is received by Xray, but there are no Watches with a **CI Integration** action defined, Xray will always respond with an indication that the build job **should** indeed fail, whether build artifacts or dependencies are found to have vulnerabilities or not.

Block Download

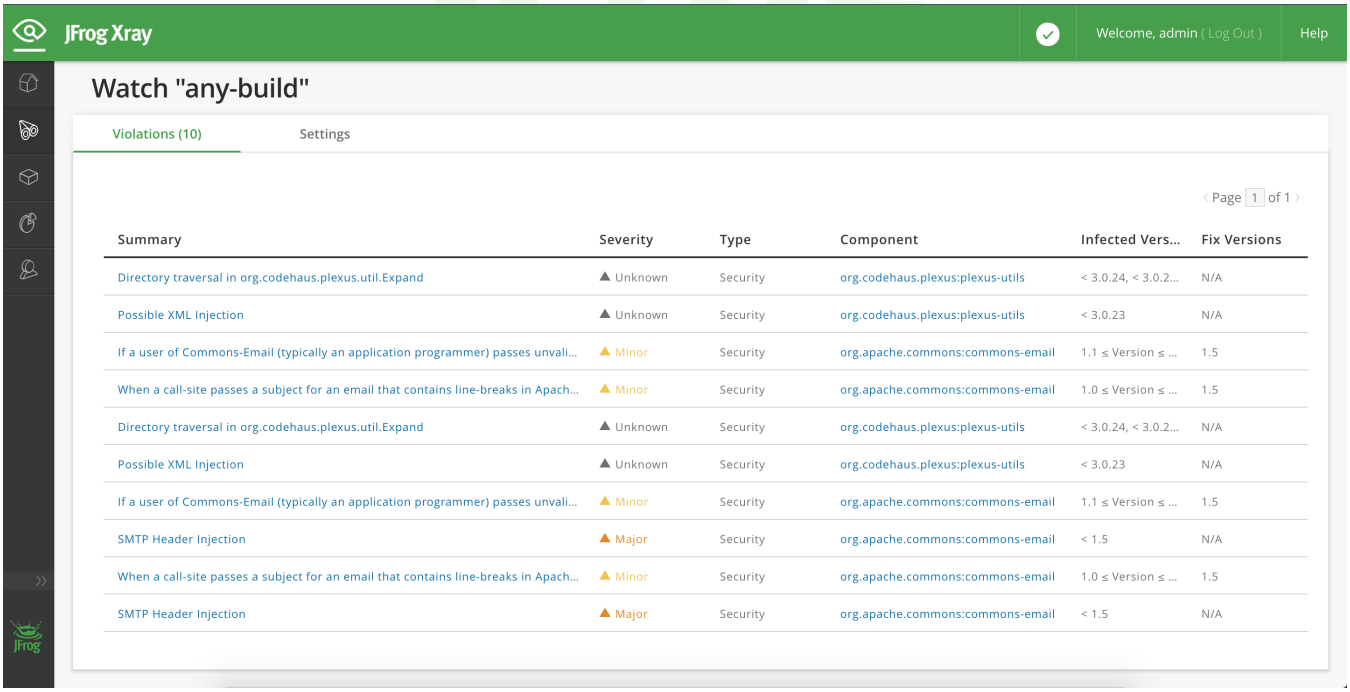
This action lets you specify that artifacts should be blocked for download from Artifactory



Block Download	When set, Artifactory will block download of artifacts that meet the Artifact Filter and Severity Filter specifications for this watch.
Block Unscanned	When set, Artifactory will block download of artifacts that meet the Artifact Filter specifications for this watch, but have not been scanned yet.

Examining Violations

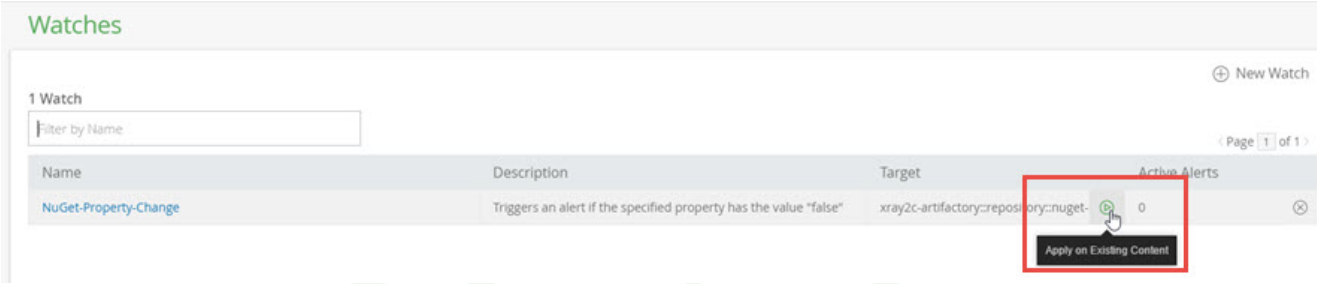
Click on a specific watch from the main Watch module page to examine all of its defined violations.



Manually Invoking a Scan

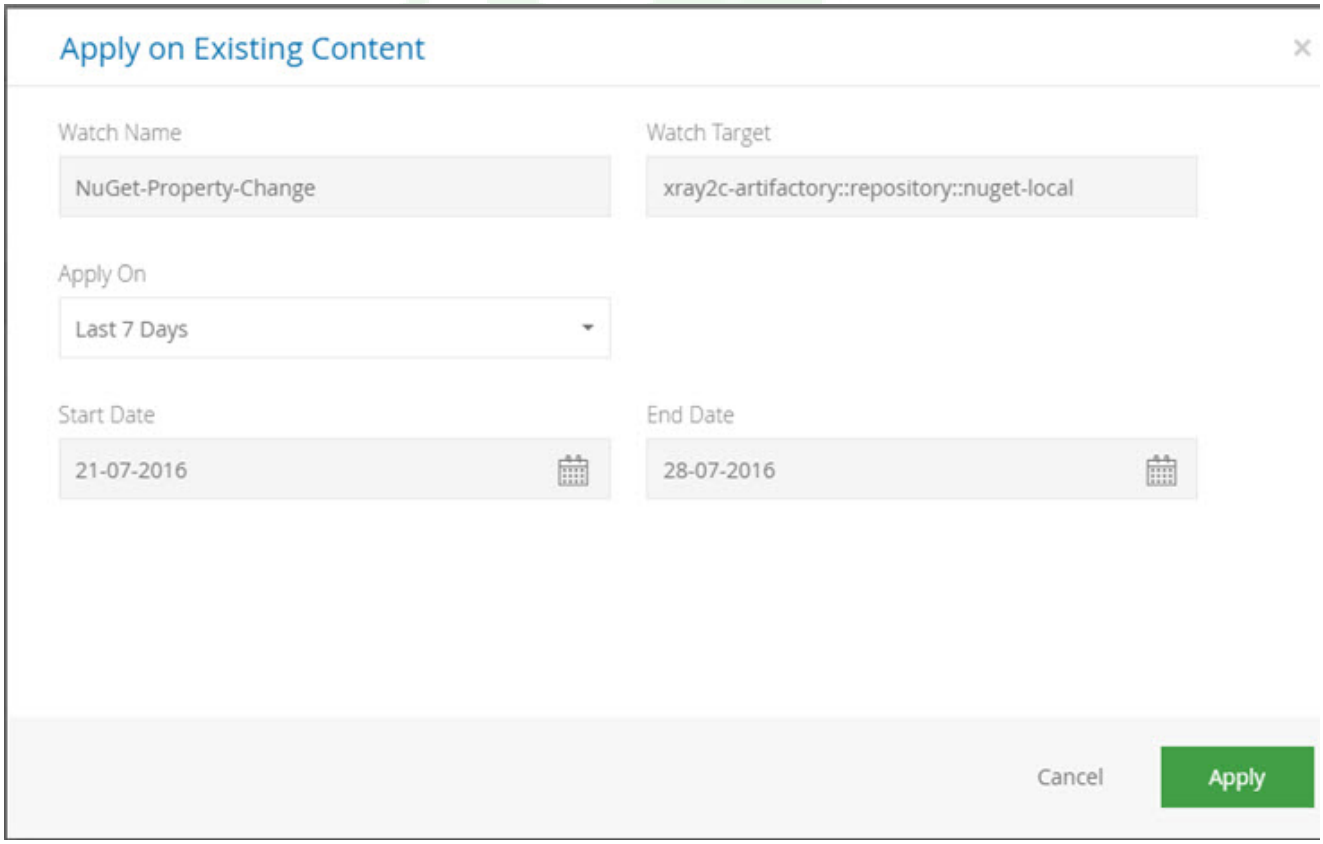
> **Temporarily Disabled**
>
> This feature has been temporarily disabled and is not available from version 1.9. The feature will be enabled again in one of the forthcoming releases.
>
> You may still initiate a scan on a specific component from the Actions Menu in its Details Panel.

Once a **Watch** is created, it will scan artifacts in the specified **Target Type** when a scan-triggering event happens, and issue alerts accordingly. However, until a scan-triggering event happens, artifacts already existing in the system will not be scanned by the watch. So, to make sure a watch is immediately applied to the relevant artifacts, you can invoke it manually by hovering over the relevant watch.



Clicking the button pops up a dialog that lets you specify a date range which defines which artifacts in the specified target type should be scanned according to the amount of time they have resided in the target.

For example, selecting "Last 7 days" will only scan artifacts that have resided in the target for the last 7 days.



Download Blocking

Previously, blocking download of artifacts was defined in Artifactory and managed as special "system watches" in Xray. From JFrog Artifactory version 5.10 and Xray version 1.12, the integration between these two applications has changed, and download blocking is now fully managed in Xray through watches that use a Block Download action.

Alerts

Overview

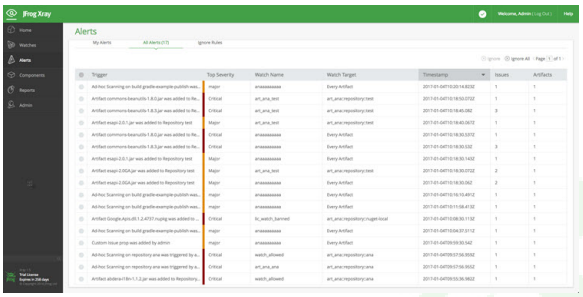
DEPRECATION NOTICE

From version 1.12, Alerts have been deprecated and are replaced by the Violations feature.

Ignore Rules can now be viewed the Watches screen. Similarly, and any Alerts present in your system when upgrading to version 1.12 are still available in the Archived Alerts tab of the Watches screen. Ignore Rules and Archived Alerts will be completely removed in forthcoming versions.

Page Contents

The **Alerts** module is where Xray displays all the alerts triggered by the different **Watches** you have defined each time it performed a scan of artifacts in the currently active Artifactory instance. Note that Xray can only provide alerts for components whose data base been registered in the Global Database Server. The main **Alerts** table displays basic information for each alert that was triggered.



Trigger	The issue that triggered the alert
Top Severity	The highest severity level reported for any issue/artifact that triggered the alert
Watch	The name of the watch that triggered the alert
Target	The target that was defined for the watch that triggered the alert
Timestamp	The timestamp of the alert
Issues	The number of issues found in the artifact that triggered the alert
Artifacts	The number of artifacts detected with the issues that triggered the alert

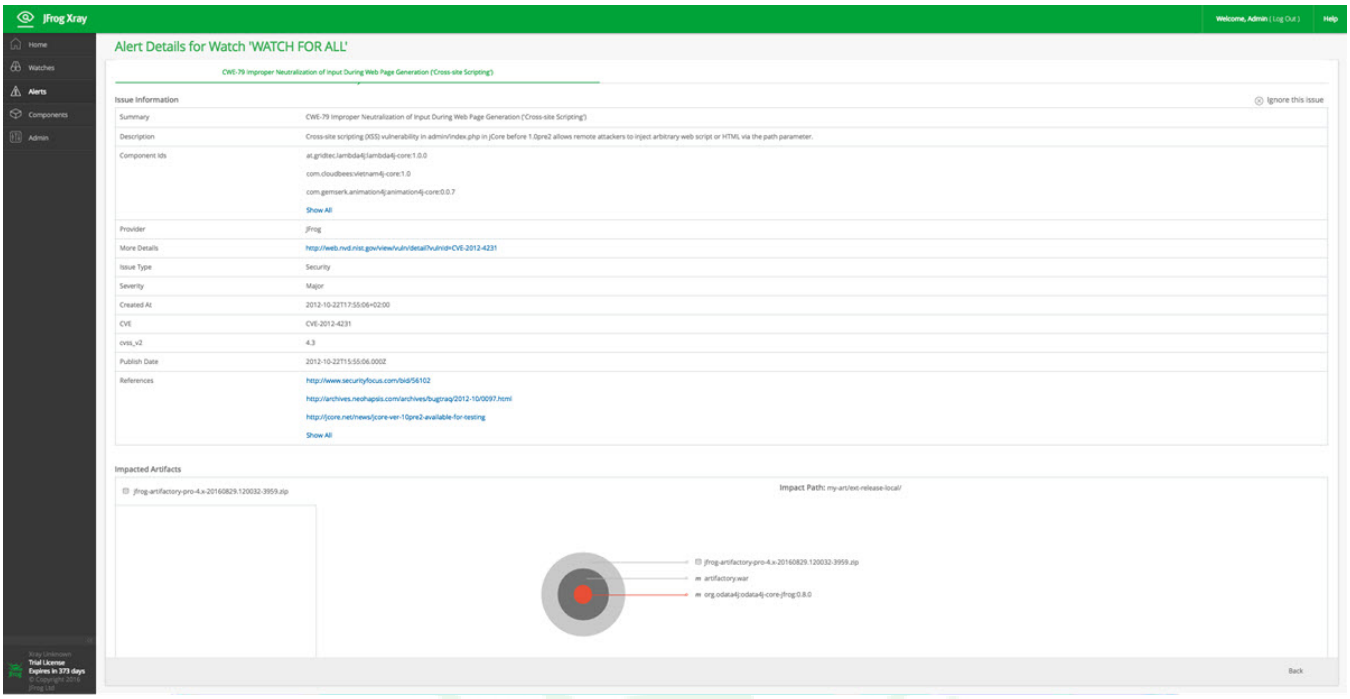
Use the **My Alerts** tab to view only the alerts triggered by watches you have defined, or the **All Alerts** tab to view all alerts in the system.

Alert Details

You can click any alert in the main alerts table to view its details.

An alert may include several issues which are displayed as horizontal tabs along the top of the screen. In addition, each issue may have an impact on several artifacts which are displayed as a set of tabs under **Impacted Artifacts.**

The screen shot below shows an alert with three issues. The first issue is selected and it impacts three different artifacts.



The screen displays the several details according to the information provided by the issue provider that reported the issue:

Summary	A title summarizing the issue.
Description	A more detailed description of the issue.
Component Ids	The Ids of the components afflicted with the issue.
Provider	The provider that identified the issue.
More Details	Additional details about the issue if available from the provider.
Issue Type	The issue type.
Severity	The issue severity.
Created At	When the issue was created.
CVE	The issue's CVE number.
cvss_v2	The issue's CVSS v2 score.
Publish Date	The date the issue was published.
References	Additional references that the issue provider may include in the stream.

Impacted Artifacts

The Impacted Artifacts panel shows all the indexed artifacts that are impacted by the selected issue. Each impacted artifact is displayed as a tab in the panel. Selecting an artifact tab displays series of concentric circles that represent the layers of the indexed artifact down to the specific artifact with the reported issue shown as a red circle in the center.

The **Impact Path** shows the path to the indexed artifact.

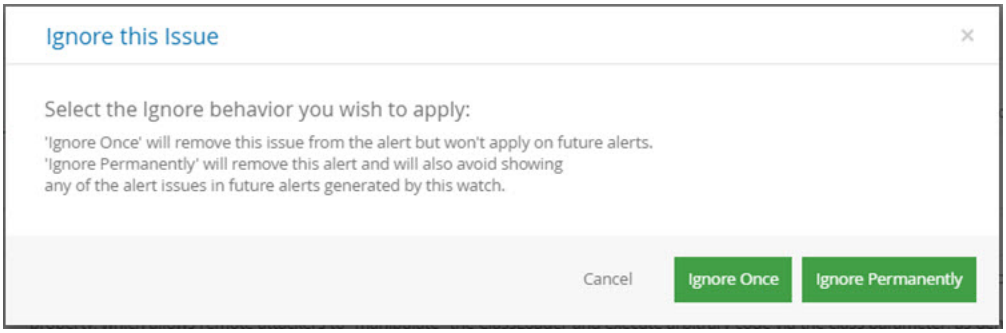
Ignoring Issues

To avoid clutter on your Alerts dashboard you can choose to ignore certain issues you may already be familiar with. To ignore an issue, click **Ignore this issue** on the Alert Details screen.

Once you select to ignore an issue you are presented with the option of:

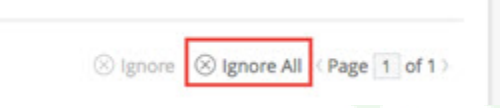
Ignore Once: The issue is only ignored in this instance of the alert

Ignore Permanently: The issue will be ignored in all future alerts. This means that if this is the only issue in an alert, the alert will not be triggered in the future.



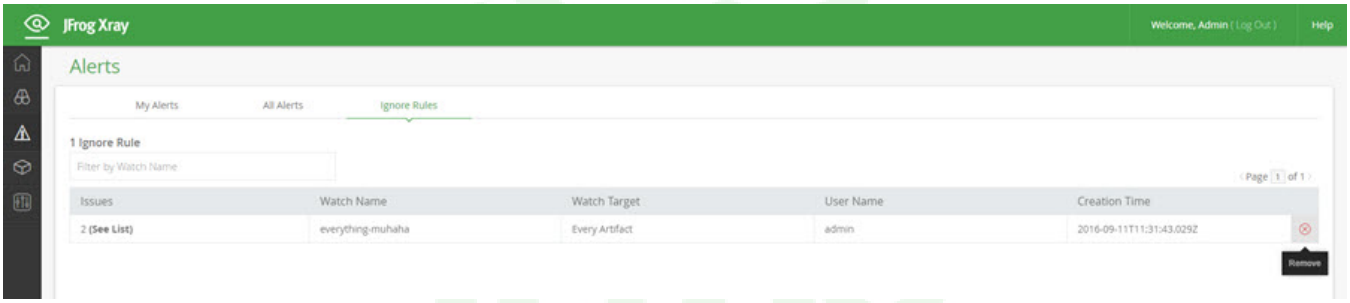
You can similarly select one or more alerts in the Alerts table and click **Ignore** to ignore all issues aggregated by the selected alerts.

Admin users also have the option to permanently ignore all alerts displayed in the screen by selecting **Ignore All.**



Ignore Rules

This tab displays the issues which you have selected to ignore permanently. If you wish to resume getting alerts for an issue that is currently being skipped due to an ignore rule, simply remove it from the list.



Watch the Screencast

Watch this screencast to learn how to use Xray's component-centric navigation.

Components

Overview

The **Components** module implements a content-driven workflow allowing you to single out relevant components you are interested in and drill down to expose greater detail so you can understand their state. This is done using the following main steps:

1. **Search**
   Enhanced search lets you single out components based on a variety of parameters.

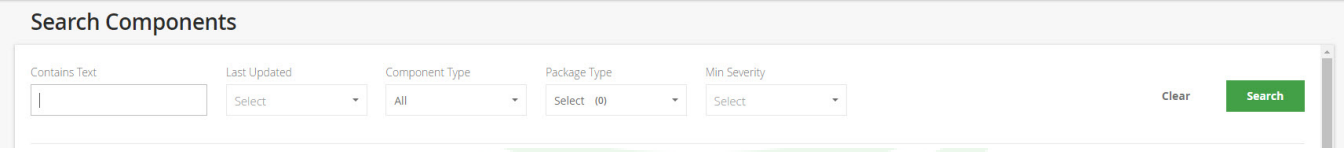
2. **Drill down**
   Once Xray has found all components that match your search query, you can select the one that interests you and drill down to get more details about it

3. **Examine violations and metadata**
   After drilling down into specific component, you can then examine all the violations detected for each version of that component and get detailed information about the violoation and about all other components in your system that are affected by it.

Page contents

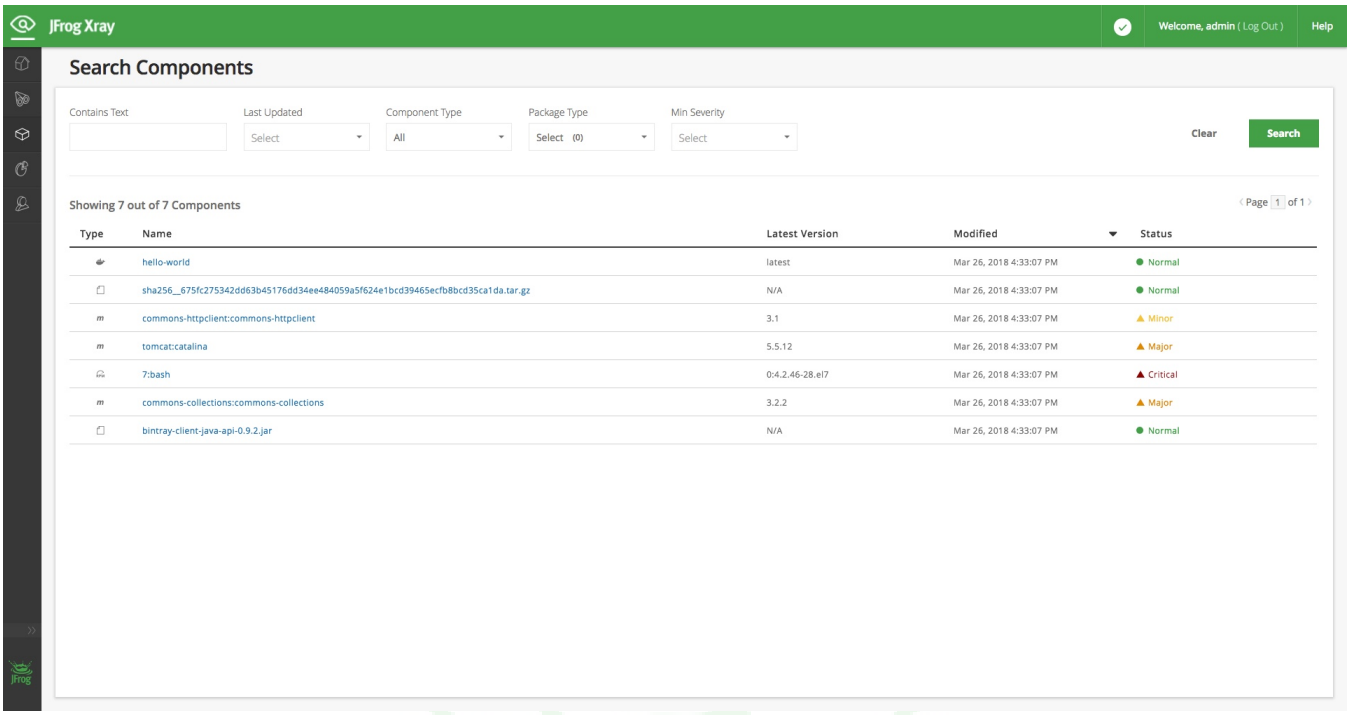
Searching for Components

At the top of the **Components** module you can enter a variety of parameters to search for specific components. Click search to run the query.



Contains Text	A free-text term to search for in the name of the component
Last Updated	Specifies when the component was last modified in Xray. You can select one of the preset time ranges, or specify a custom range.
Component Type	Specifies whether you are searching for a Package, a Build or a File or
Package Type	Restricts search results to the specified package type
Min Severity	Only components with vulnerabilities with the specified severity and above will be displayed

Search Results



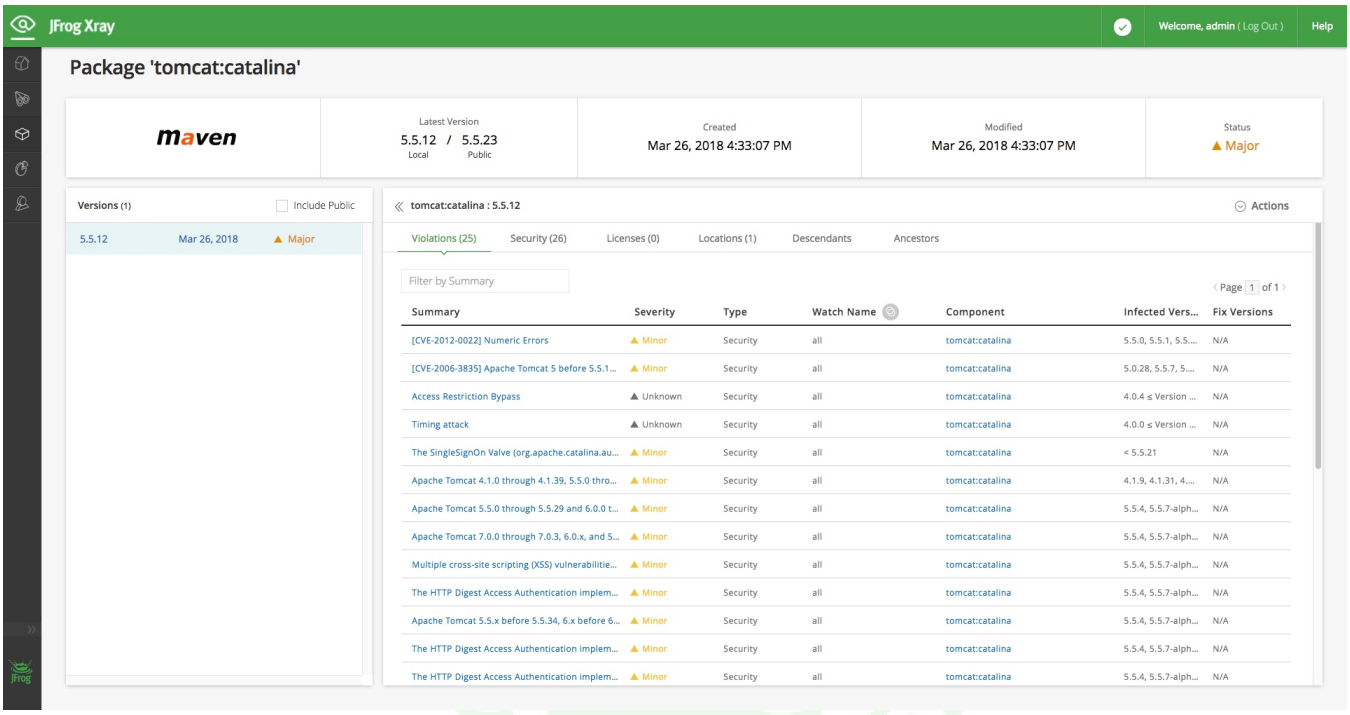
The search results are displayed in a table showing the following parameters

Type	Indicates if the component is a package, a build or a file
Name	The name of the component
Latest Version	The latest version of the component where applicable ("files" don't have versions)
Last Updated	Indicates when the component was last modified in Xray (e.g., last indexed or status changed)
Issues	The number of issues detected in the component
Status	Indicates the highest severity of any of the issues found for the component. "Normal" means no issues were found.

Component Details

To drill down and view the details about a component, click its name in the list of search results. The Component Details view is split up into three panels:

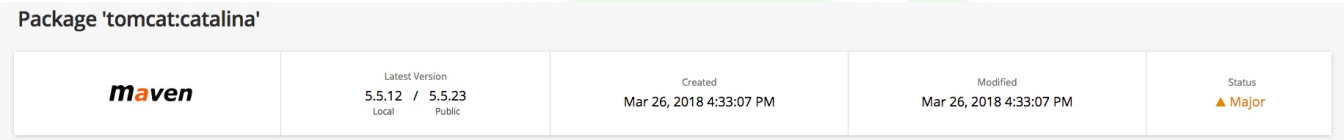
- Summary Strip
- Versions Panel
- Details Panel



Summary Strip

The strip at the top of the Component Details view varies slightly depending on whether the component is a package, a build or a file, and displays a summary of the components most basic information.

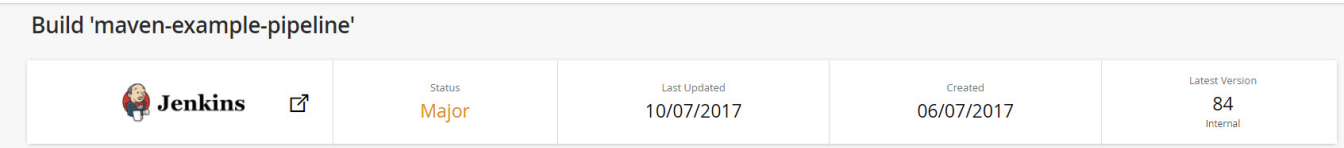
Package



For a package, the summary strip displays:

- The package type logo for quick and easy identification
- **Latest Version:** The latest version of the package that is available. The "Internal" version shows the latest version that is hosted by your Artifactory instance, and "Public" shows the latest version that is publicly available on the external web.
- **Created: The package's creation date**
- **Last Updated:** Last time the package was indexed or modified
- **Status:** The highest severity of any vulnerability found in the package

Build



For a build, the summary strip displays:

- The logo of the CI server that ran the build with a link for direct and easy access to the build in Artifactory
- **Status:** The highest severity of any vulnerability found in the build
- **Last Updated:** Last time the build was indexed or modified
- **Created:** The build's creation date
- **Latest Version:** The latest version of the build that is available.

File

For a file, the summary strip displays:

- A file icon
- **Status:** The higher of the highest severity watch violation and highest severity of any vulnerability found in the file
- **Last Updated:** Last time the file was indexed or modified
- **Created:** The file's creation date

Versions Panel

The **Versions** panel displays all the versions of the selected component that have been indexed by Xray. Select any of these versions to display detailed information about them. If publicly available versions of the selected component are available, Xray will display the **Include Public** checkbox. When set, Xray will also display those versions in the list, however, note that when selecting one of these versions, Xray may not be able to display additional information.

> **Specific Versions**
>
> Select any version displayed in the Versions panel to get a list of issues detected in that specific version.

Details Panel

The details panel displays several details about the selected component including:

tomcat:catalina : 5.5.12 Actions

Violations (25) Security (26) Licenses (0) Locations (1) Descendants Ancestors

Filter by Summary

Page 1 of 1

Summary	Severity	Type	Watch Name	Component	Infected Vers...	Fix Versions
The Windows installer for Apache Tomcat 6.0.0 ...	Major	Security	all	tomcat:catalina	5.5.0 ≤ Version ...	N/A
[CVE-2012-0022] Numeric Errors	Minor	Security	all	tomcat:catalina	5.5.0, 5.5.1, 5.5....	N/A
[CVE-2006-3835] Apache Tomcat 5 before 5.5.1...	Minor	Security	all	tomcat:catalina	5.0.28, 5.5.7, 5....	N/A
The SingleSignOn Valve (org.apache.catalina.au...	Minor	Security	all	tomcat:catalina	< 5.5.21	N/A
Apache Tomcat 4.1.0 through 4.1.39, 5.5.0 thro...	Minor	Security	all	tomcat:catalina	4.1.9, 4.1.31, 4....	N/A
Apache Tomcat 5.5.0 through 5.5.29 and 6.0.0 t...	Minor	Security	all	tomcat:catalina	5.5.4, 5.5.7-alph...	N/A
Apache Tomcat 7.0.0 through 7.0.3, 6.0.x, and 5...	Minor	Security	all	tomcat:catalina	5.5.4, 5.5.7-alph...	N/A
Multiple cross-site scripting (XSS) vulnerabilitie...	Minor	Security	all	tomcat:catalina	5.5.4, 5.5.7-alph...	N/A
The HTTP Digest Access Authentication implem...	Minor	Security	all	tomcat:catalina	5.5.4, 5.5.7-alph...	N/A
Apache Tomcat 5.5.x before 5.5.34, 6.x before 6...	Minor	Security	all	tomcat:catalina	5.5.4, 5.5.7-alph...	N/A
The HTTP Digest Access Authentication implem...	Minor	Security	all	tomcat:catalina	5.5.4, 5.5.7-alph...	N/A
The HTTP Digest Access Authentication implem...	Minor	Security	all	tomcat:catalina	5.5.4, 5.5.7-alph...	N/A
DigestAuthenticator.java in the HTTP Digest Ac...	Minor	Security	all	tomcat:catalina	5.5.4, 5.5.7-alph...	N/A

- **Violations:** These are violations to filters defined on a watch. They are only reported for the root component, not for its dependencies.
- **Security:** Known security vulnerabiliites for the selected component
- **Licenses:** OSS licenses used by the component
- **Locations:** Locations where the files of the component can be found
- **Descendants:** Components that the selected component includes (depends on)
- **Ancestors:** Components that include (depend on) the selected component

To focus on specific violations, you may filter the list displayed using the **Filter by Summary** field.

Infected Versions

The **Violations** tab of the Details panel provides the set of versions that are infected with the violation. The set can include a range of versions and specific versions in any combination. For example, "`2.0ga, 2.0_rc9, 2.0_rc10, 2.0_rc11, 2.0.1, 2.1.0 version 2.1.0.1`".

Remediation

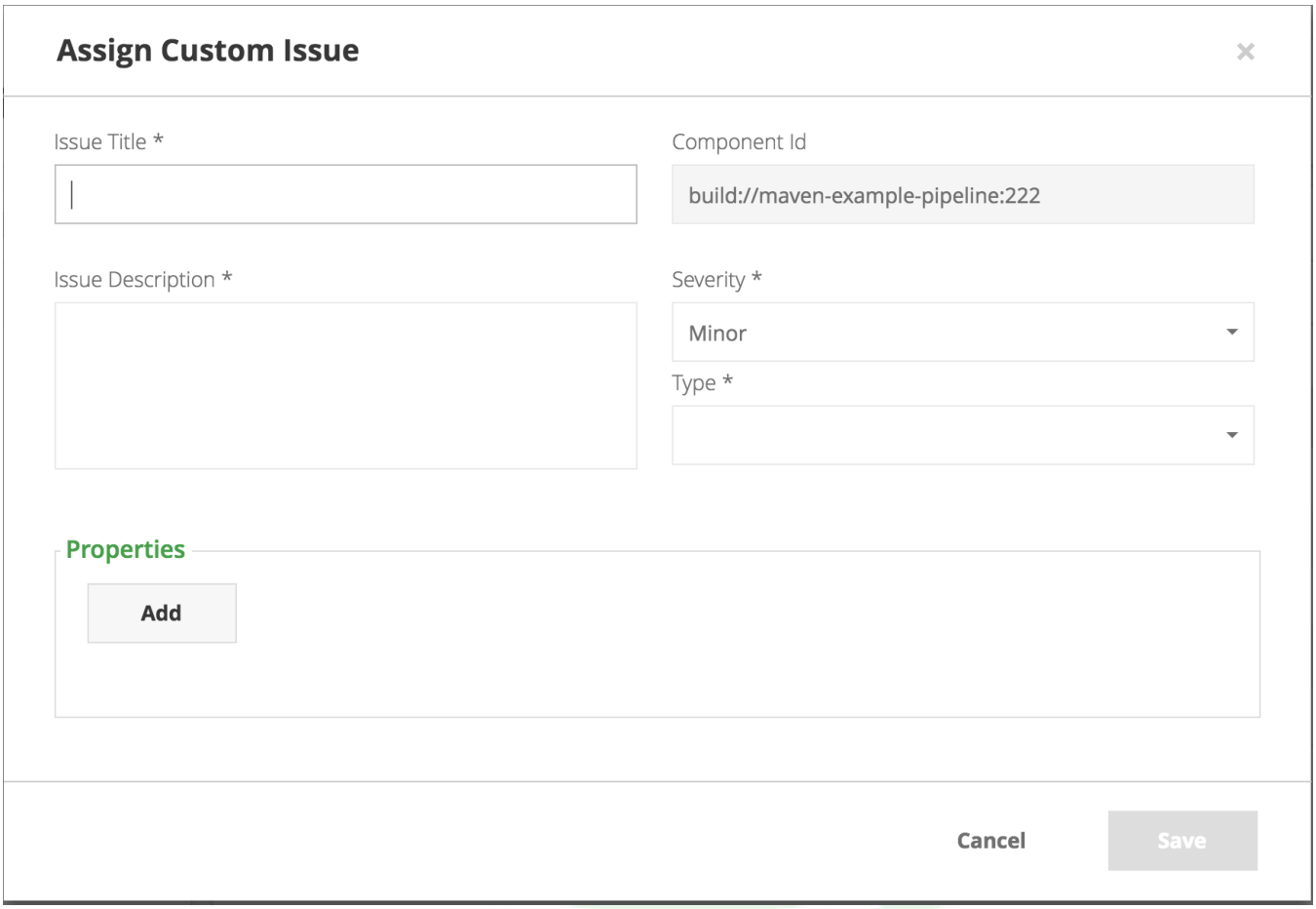
The Fix Versions tab of the Details panel provides remediation information for the violation. This field indicates in which version of the selected components the violation has been fixed giving you the opportunity to upgrade to that version and thus remedy the violation.

Actions Menu

The Actions menu in the Details panel lets you perform the following actions on the selected component:

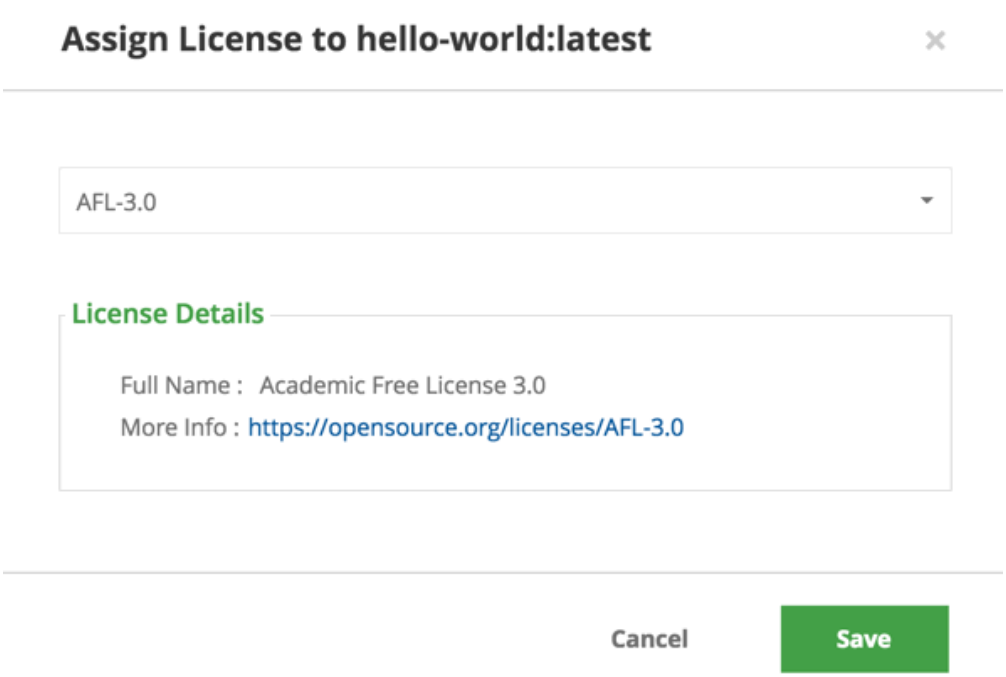
Scan for Violations: Scans the current component for violations

Assign Custom Issue: Lets you specify a custom issue and assign it to the component:



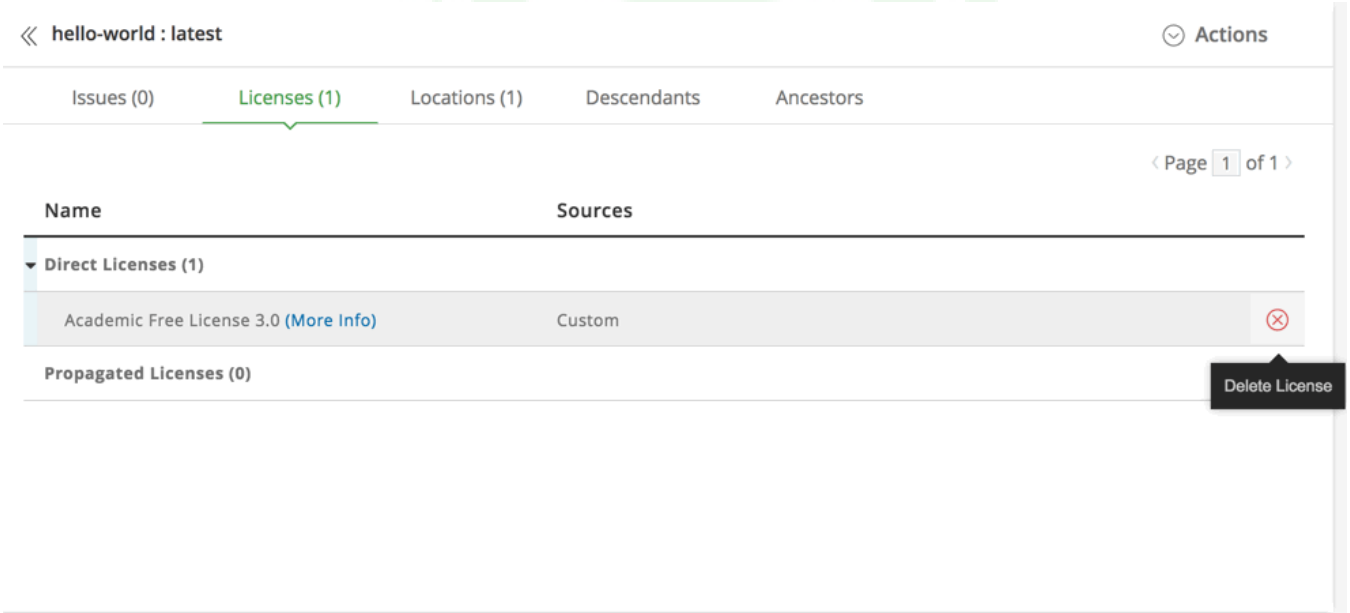
Issue Title	A descriptive title for the issue.
Component ID	The ID of the component to which the issue was assigned.
Issue Description	A more description of the issue.
Severity	The issue severity
Type	The issue type
Properties	Allows you to add custom properties to the issue

Assign a Custom License: Lets you assign a custom license to a component:



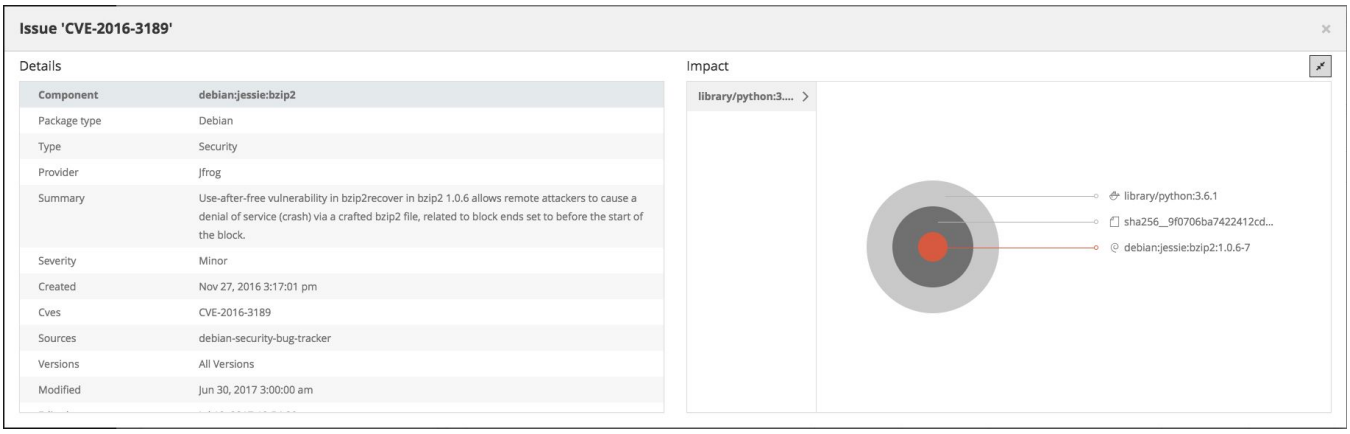
A license created by a user is tagged as a Custom license and can be deleted by users assigned with the Manage Components permission. The custom license is assigned to a specific version and is propagated to parent components and is part of their license list. It triggers an impact analysis and generates violoations in case it matches criteria of any existing Watches.

The new license is included in the scan the next time a security report is generated.



Examining Violations

To examine the details of a violation, click the violoation in the list displayed on the Component Details panel to display the Violoation Details popup.



The **Impact** panel of the Violoation Details popup provides a list of all components which are impacted by this violation. Select any component in the list to view the full hierarchy of components affected.

Watch the Screencast

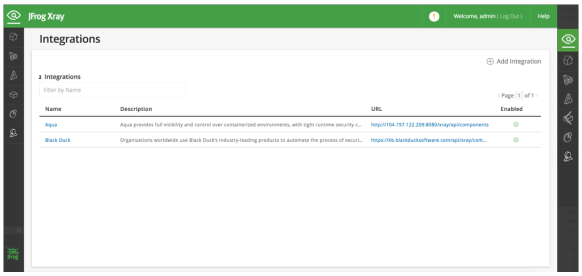
Watch this screencast to learn how to use Xray's component-centric navigation.

Integrations

Overview

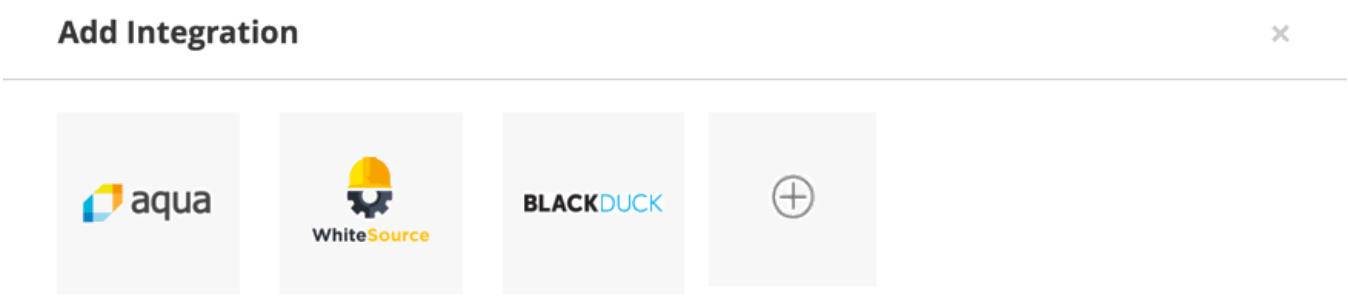
Page Contents

JFrog Xray is open for integration with any number of issue and vulnerability providers and pre-configured with a number of providers out-of-the-box. In addition, you can connect to additional issue and vulnerability feeds if you have accounts with the corresponding providers. The **Integrations** screen in the **Admin** module displays the integrations you have configured and connected to.



To add a new integration, click "Add Integration". The button will be disabled if all integrations currently available in the system have already been configured.

The **Add Integration** dialog shows all available integrations you can connect to, and provides the option to add a custom integration. Select the provider you wish to connect to from the list of icons displayed, or click the plus sign to add a custom integration.

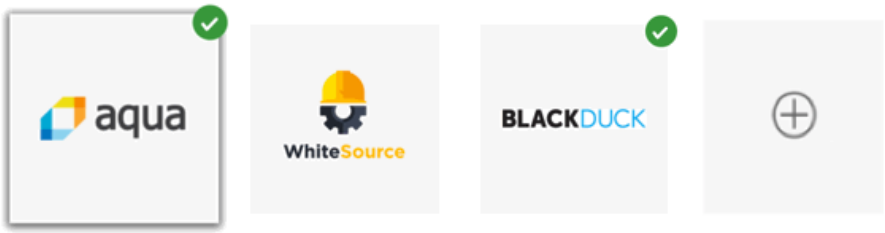


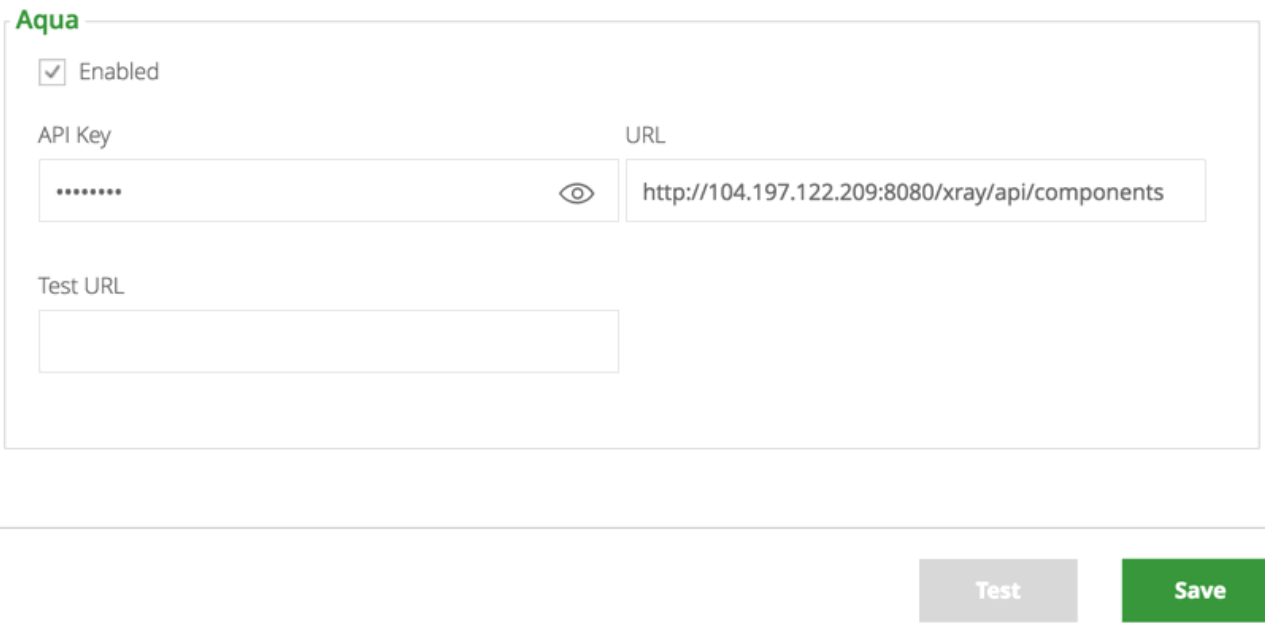
To connect to a provider, set the **Enabled** checkbox and enter the following parameters:

- The **API Key** you received from the provider
- The **Test URL** you can use to test your API key with the provider using the "*Test*" button.
- The **URL** Xray uses to check if a component it is scanning is registered with the provider.

Aqua

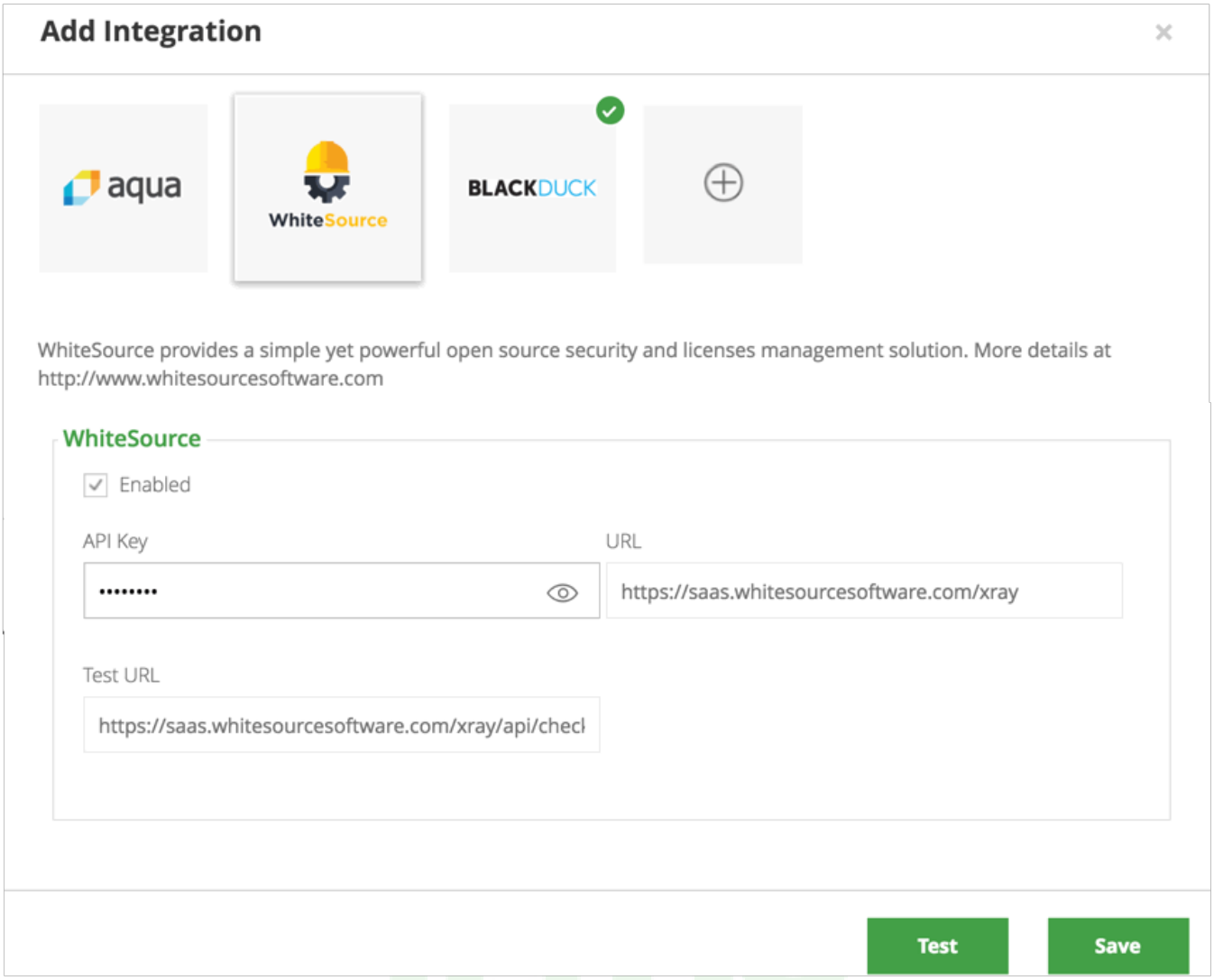
Aqua Security offers a comprehensive security solution for containerized environments. If you have an account with Aqua, you may enable this feed, enter your Aqua API key, the URL of your on-prem Aqua installation and the test URL.





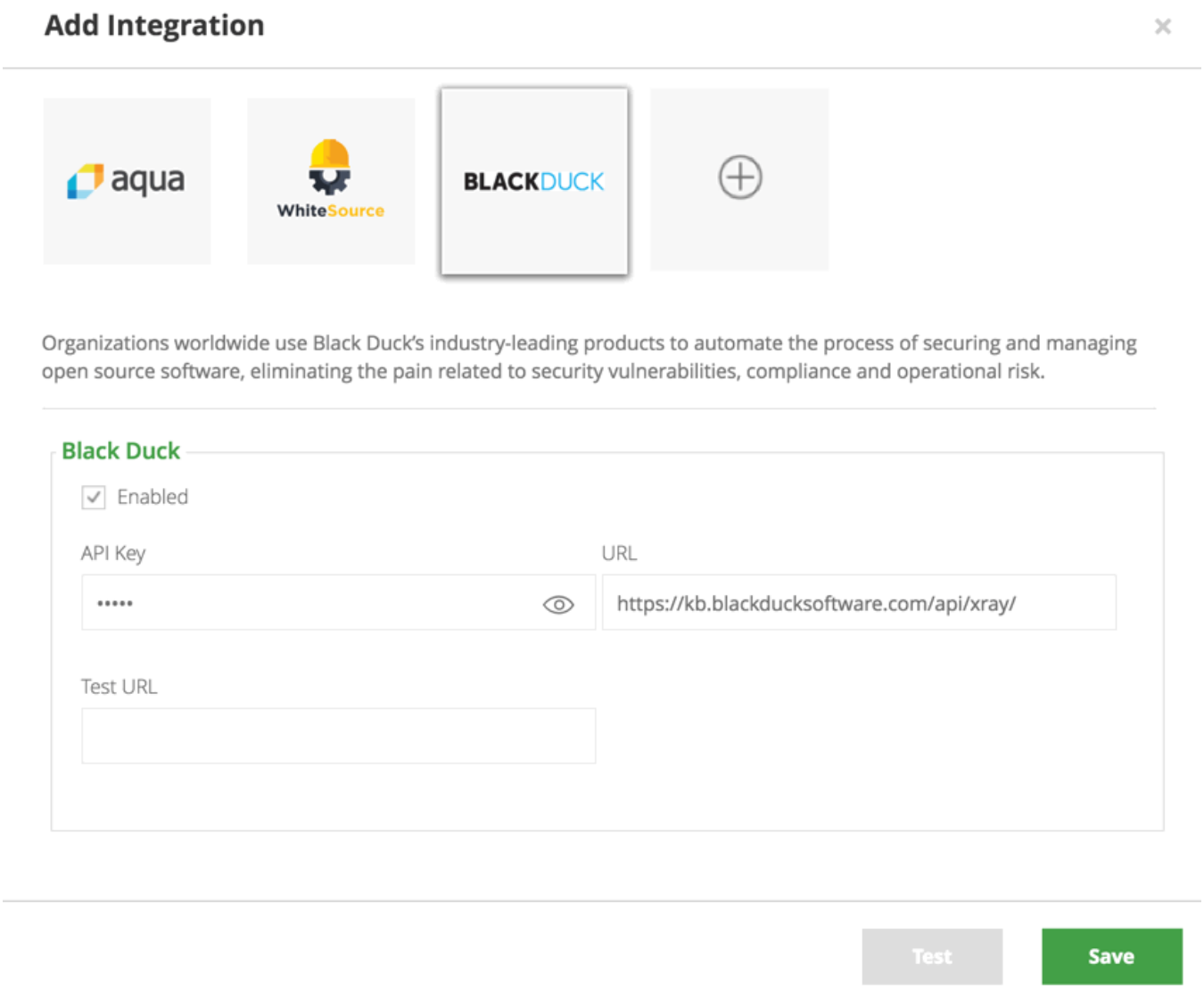
WhiteSource

WhiteSource offers a security and license management solution for your open source components. If you have an account with WhiteSource, you may enable this feed and enter your WhiteSource API key, URL and Test URL.



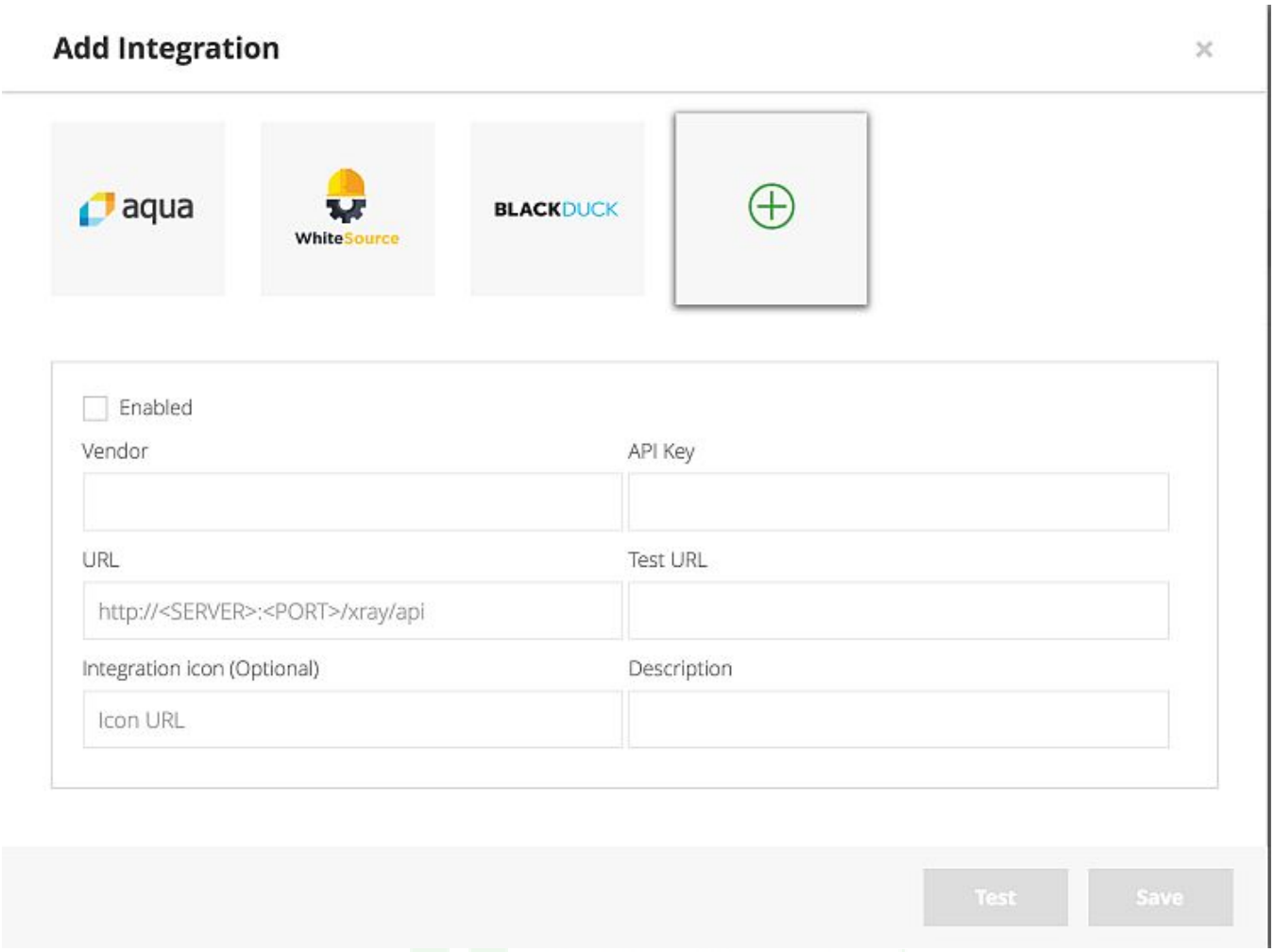
Black Duck

Black Duck offers an enterprise-grade solution to automate the process of securing, managing, and ensuring license compliance for open source software in applications and containers. If you are a Black Duck customer, you may enable this feed by purchasing the standard Hub edition with the security module, enter the provided Black Duck API key, the URL of your Black Duck installation and the test URL to start using Black Duck data within Xray.



Adding a Custom Integration

In addition to the integrations included out-of-the-box, Xray also allows you to create custom integrations. This gives you the opportunity to add analyses from different providers with whom you may have an account, or even to create your own provider and display information such as performance issues, known defects or any other information offered by your provider.



To add and connect to a custom provider, set the **Enabled** checkbox and enter the following parameters:

- The **Vendor** name
- The **API Key** you received from the provider
- The **URL** Xray uses to check if a component it is scanning is registered with the provider.
- The **Test URL** you can use to test your API key with the provider using the "*Test*" button.
- The **URL to an icon** you can optionally display for the vendor
- A **Description** for the vendor

Reports

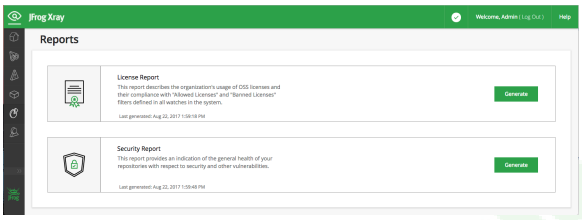
Page contents:

Overview

The **Reports** module lets you display the different reports available in the system. Currently, there are two reports you can run:

License Report	Provides information about the distribution of open source licenses used by components indexed by Xray, and their compliance according to `Allowed Licenses` and `Banned Licenses` filters you have defined in the system.
Security Report	Indicates the general health of your repositories with respect to security and other vulnerabilities.

Xray analyzes indexed artifacts and runs all reports automatically every few minutes in the background. When you click **Generate** for any of the available reports, Xray displays the data cached from the last run.

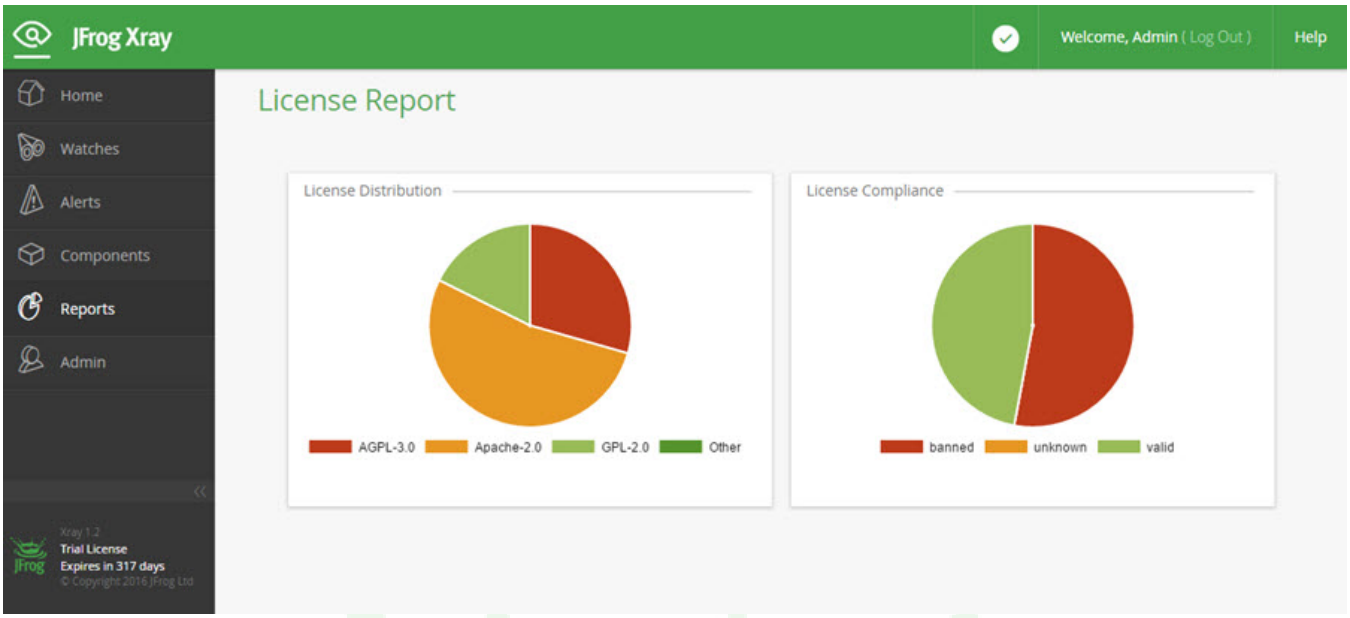


> **Synchronized the Database**
>
> Note that you can only generate reports after synchronizing with the Global Database Server at least once.

License Report

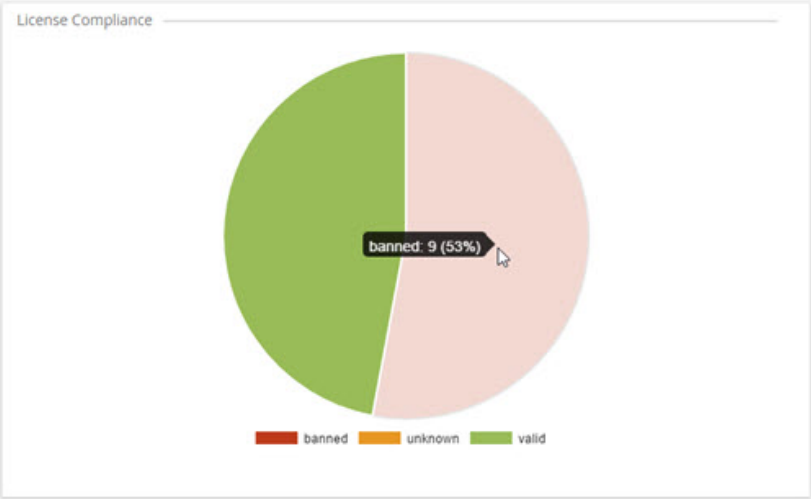
The License Report displays two main readings:

License Distribution	The **License Distribution** chart displays the distribution of licenses found in all artifacts indexed in the system. Note that only license types that make up at least 5% of the total distribution are displayed in a separate segment. Any license type with less than 5% distribution is accumulated into an "Other" category
License Compliance	The **License Compliance** chart displays the compliance of licenses found in the system according to "Allowed Licenses" and "Banned Licenses" filters defined in all watches in the system.



Segment Details

Hovering over any segment displays the number of artifacts that make up that segment and its percentage of the whole.



Clicking on any segment (or the corresponding item in the chart's legend) displays the list of components that make up that segment.

License Distribution : MIT

Filter by Component Name

Page 1 of 3

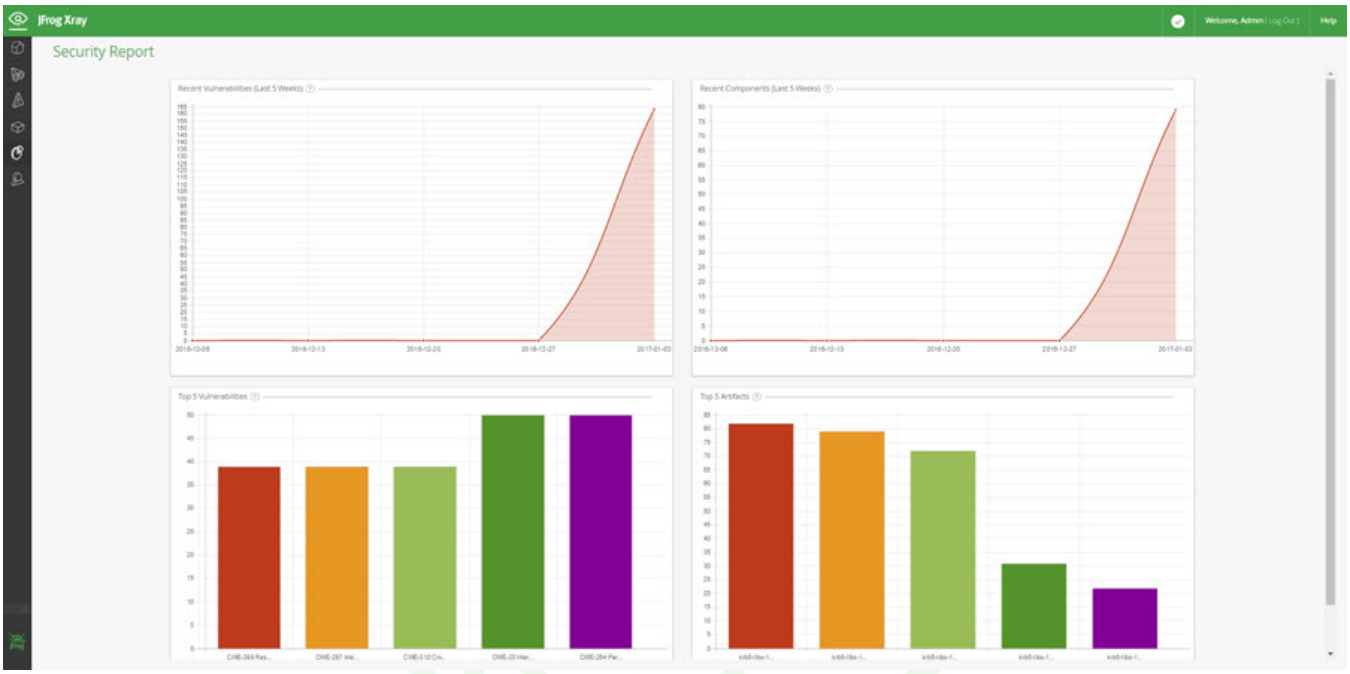
Component Name	Package Type
chalk	Npm
chalk	Npm
chalk	Npm
chalk	Npm
chalk	Npm
ansi-regex	Npm
ansi-regex	Npm
ansi-regex	Npm
ansi-styles	Npm
ansi-styles	Npm

Click on any component to view its details in the Components module.

Security Report

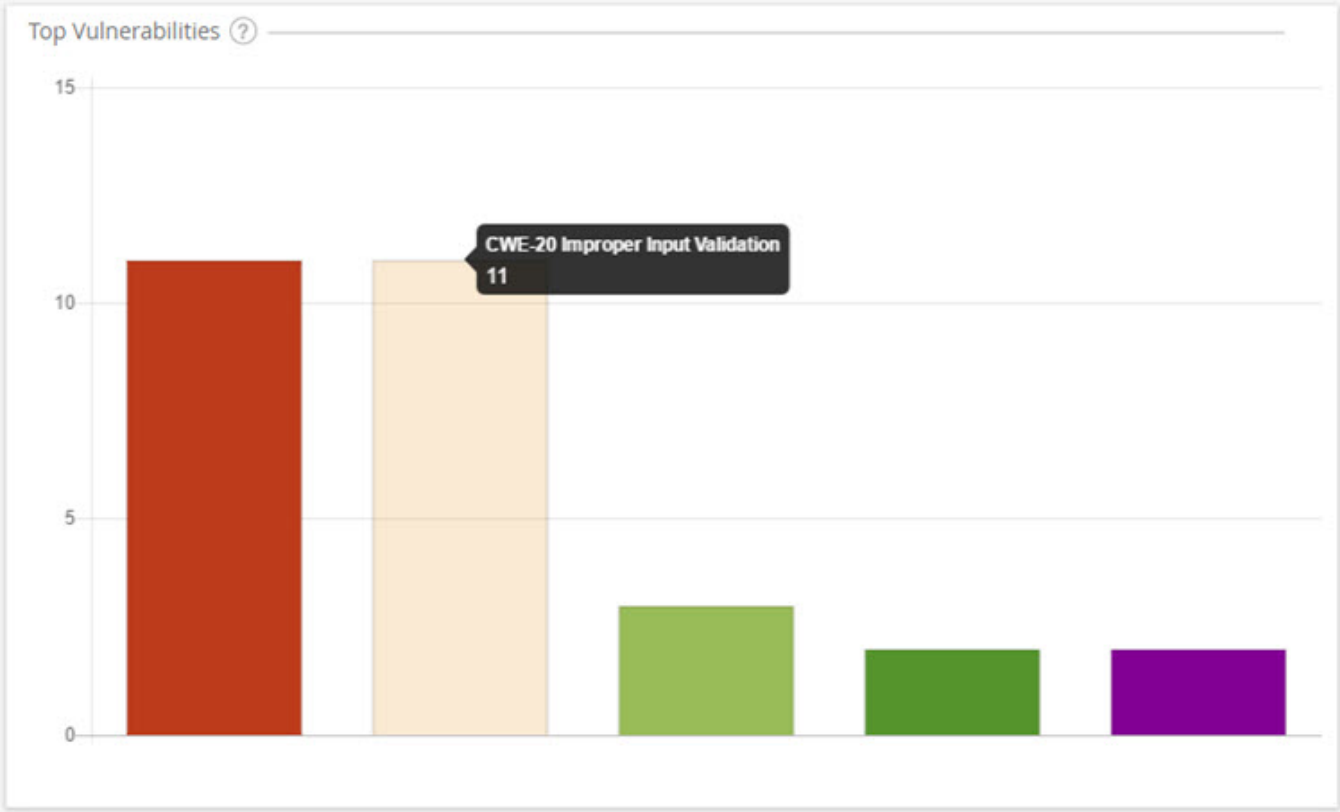
The Security Report gives you an indication of the general health of your repositories with regard to security vulnerabilities. Once generated, you can view the following charts:

Recent Vulnerabilities	Shows the vulnerabilities that were most recently detected in components that Xray has indexed.
Recent Components	Shows the components indexed by Xray that were most recently detected to include vulnerabilities.
Top Vulnerabilities	Shows the vulnerabilities that have the most wide-reaching effects on your repositories in that they are included as dependencies by the largest number of components indexed by Xray.
Top Artifacts	Shows the artifacts indexed by Xray that were detected to have the largest number of vulnerabilities, either directly or as a result of included dependencies.



Bar Details

Hovering over a bar or data point in any chart provides additional information. For example, hovering over a bar in the Top Vulnerabilities chart displays the vulnerability that affects the largest number of components indexed by Xray, and the number of components affected



Clicking on the bar displays full details. For example clicking a bar in the Top Vulnerabilities chart shows full details for vulnerability that affects the largest number of components indexed by Xray.

Top Vulnerabilities

Summary	CWE-20 Improper Input Validation
Description	The wsdl_first_https sample code in distribution/src/main/release/samples/wsdl_first_https/src/main/ in Apache CXF, possibly 2.6.0, does not verify that the server hostname matches a domain name in the subject's Common Name (CN) or subjectAltName field of the X.509 certificate, which allows man-in-the-middle attackers to spoof SSL servers via an arbitrary valid certificate.
Severity	Major
Properties	CVE : CVE-2012-5786 CVSS_V2 : 5.8
Created	04-11-2012

CI-CD Integration

Overview

Failing a build job that includes build artifacts or dependencies with vulnerabilities is an effective way to prevent any infected builds from reaching your production systems. There are organization policies that force developers to scan every build they run and fail them immediately if infected artifacts are found. However, this mode of operation has been found to inhibit developers' creativity and stunt their productivity, and often, developers find a way around this kind of restriction. A better solution is to periodically run this kind of scan once the code of several developers has been merged. For example, during a nightly build run by a organization's CI server.

JFrog Xray can be integrated into an organization's CI/CD pipeline to make sure that build jobs containing vulnerabilities are stopped early on in the process. As part of a fully automated process, Xray receives information about a build that has just been run by your CI server, it then runs a deep recursive scan on the build down to the deepest level dependency, and if any vulnerabilities are found, Xray will return an indication to the calling CI server.

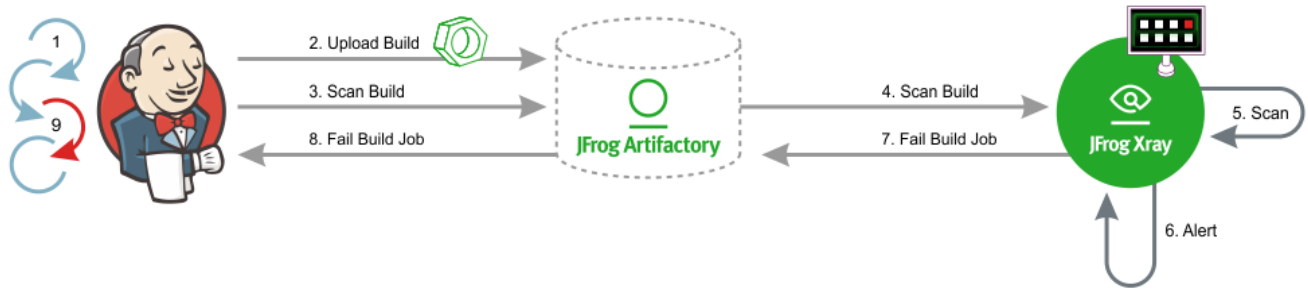
Page contents

The Process

There are three players in this process:

- **Your CI Server** (currently, Jenkins CI and TeamCity are supported)
  The CI server runs a build job and sends a request to Xray, through Artifactory, for the build to be scanned. If the scan detects a vulnerability, the CI server can take appropriate action as configured in the build job.
- **JFrog Artifactory**
  JFrog Artifactory serves as a mediator between the CI server and Xray. It does nothing more than pass information between one and the other.
- **JFrog Xray**
  Upon request, if Xray has defined watches with **Actions** to fail a build job, it will scan the build, and respond with a message that the build job should fail if a vulnerability is detected in the build artifact or one of its dependencies.

The diagram below illustrates how the process is implemented using Jenkins CI.



1. Jenkins runs a build job.
2. Assuming the build is successful, Jenkins uploads the build to Artifactory. New build artifacts and dependencies are automatically indexed by Xray.
3. Jenkins passes a request to Artifactory to scan the build.
4. Artifactory passes a request to scan the build through Xray's **scanBuild** REST API endpoint.
5. Xray scans the build according to a defined Watch with a **Fail Build Job** Action.

> **Multiple watches or no watches?**
>
> You may define multiple Watches with a Fail Build Job Action, each with its own criteria (i.e. Artifact Filters and/or Issue Filters) that should trigger an alert. All of these Watches are applied each time a build is scanned.
>
> If Xray receives a scanBuild request, and there are **no** Watches defined with a Fail Build Job Action, Xray will always respond with an indication to fail the build job, even if no vulnerabilities are found in the build artifacts or their dependencies.

6. If any build artifact or dependency meets the conditions (filters) defined in the Watch, Xray triggers an alert and...
7. Xray responds to the **scanBuild** request indicating that the build job should fail.

> **All Alerts in one response**
>
> The response includes the details of all Alerts generated by all Watches that include a Fail Build Job Action.

8. Artifactory passes on the response back to Jenkins.
9. Jenkins fails the build job.

The Result

Xray's build integration allows you to manage your build jobs and configure them with appropriate actions if build artifacts or dependencies with vulnerabilities are found in your builds. While the default action (in Jenkins) is to simply stop the build, you can actually configure your pipeline to do other things like send email notifications or even run a different build job.

Configuration

Configuring Xray

Xray supports CI/CD integration from **version 1.6**

For Xray to scan builds upon request by a CI server, you need to configure a **Watch** with the right filters that specify which artifacts and vulnerabilities should trigger an alert, and set a Fail Build Job Action for that Watch.

Configuring your CI Server

Xray CI/CD integration is supported for Jenkins CI and TeamCity.

Jenkins

To **configure a build job** to request a scan, with the Jenkins Artifactory Plugin (v2.9.0 and above), you need to create a `scanConfig` instance and and pass it to the `xrayScan` method in the Jenkins Pipeline.

TeamCity

To **scan build artifacts and dependencies** for vulnerabilities, with the TeamCity Artifactory Plugin, you need to enable the **Xray scan on build** and **Fail build options**, configured per build.

Configuring Artifactory

While Artifactory does not play an active part in this integration, and there is no explicit configuration needed, Artifactory does play a passive role in passing information between your CI server and JFrog Xray.

This feature is supported in **Artifactory from v4.16** and above.

Watch the Screencast

Watch this screencast to learn how to get the best of two worlds - developer productivity and safety, by scanning the results of every build for security vulnerabilities, license compliance issues and more with JFrog Xray.

IDE Integration

Overview

The cost of remediating a vulnerability is akin to the cost of fixing a bug. The earlier you remediate a vulnerability in the release cycle, the lower the cost.

JFrog Xray is instrumental in flagging components when vulnerabilities are discovered in production systems at **runtime**, and also, through integration to CI systems like Jenkins CI and TeamCity at **build time**. The IntelliJ IDE integration completes the CI/CD process, by bringing Xray's issue discovery one step earlier, to **development time**.

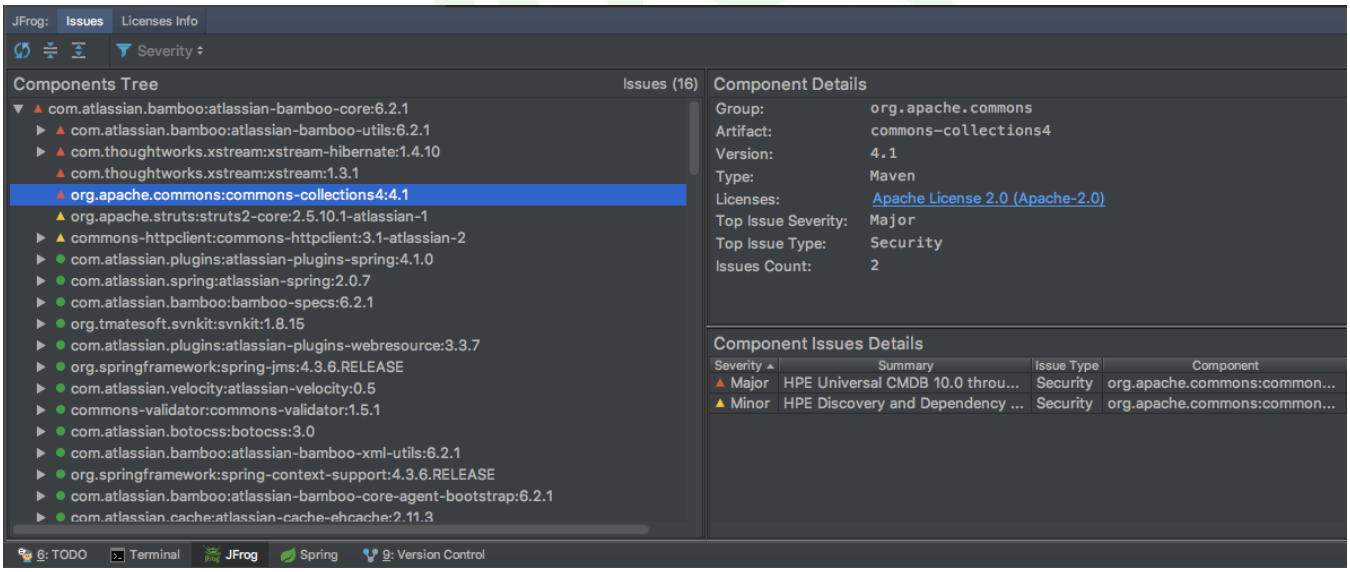
Currently, the JFrog IntelliJ IDEA plugin supports Maven, Gradle and npm components, but coverage will be extended to additional industry-standard IDEs and to additional package formats.

Page contents

JFrog IntelliJ IDEA Plugin

> From JFrog Xray **version 1.9**, IntelliJ IDEA users connecting to Xray from Intellij are required to be granted the 'View Components' action in Xray. To learn more about Xray actions, see the Actions section.

The JFrog IntelliJ IDEA plugin adds JFrog Xray scanning of Maven, Gradle, and npm project dependencies to your IntelliJ IDEA. It allows developers to view panels displaying vulnerability information about the components and their dependencies directly in their IntelliJ IDEA. With this information, a developer can make an informed decision on whether to use a component or not before it gets entrenched into the organization's product. The plugin filter allows you view the scanned results according to issues or licenses.



Installation and Setup

To install and work with the plugin, perform these actions:

1. Install the JFrog plugin, using one of these options:
    - Install from the IntelliJ plugin repository.
    - Install Plugin from Disk: Download from Bintray or create the plugin from sources.
2. Configure the Plugin to Connect to JFrog Xray.
3. Scan and view the results.
4. Filter Xray Scanned Results.

You need to make sure your system meets the prerequisites listed below.

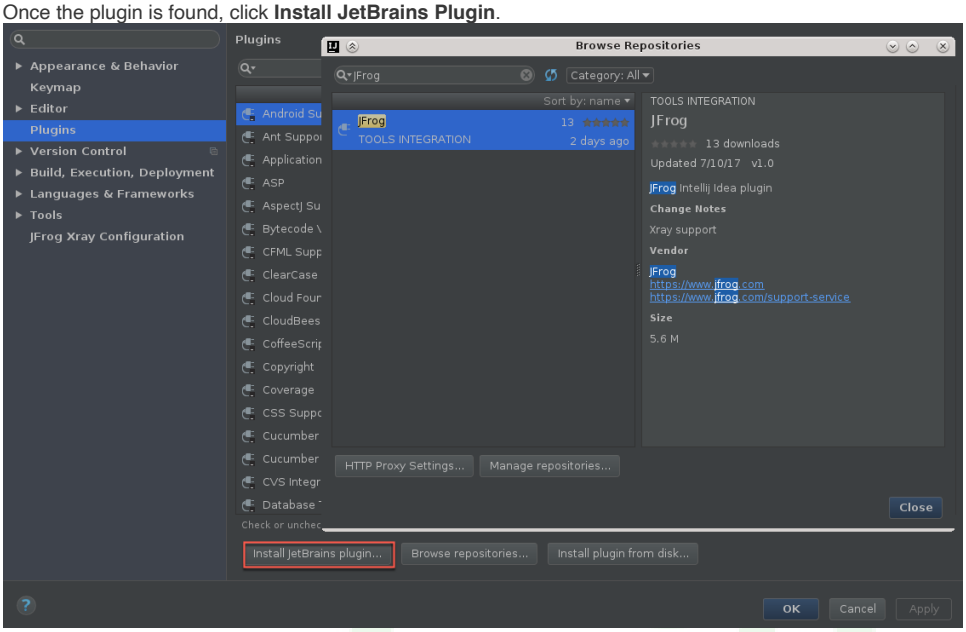
Prerequisites

IntelliJ IDEA version 2016.2 and above.

JFrog Xray version 1.7.2.3 and above.

Installing from the IntelliJ Plugin Repository

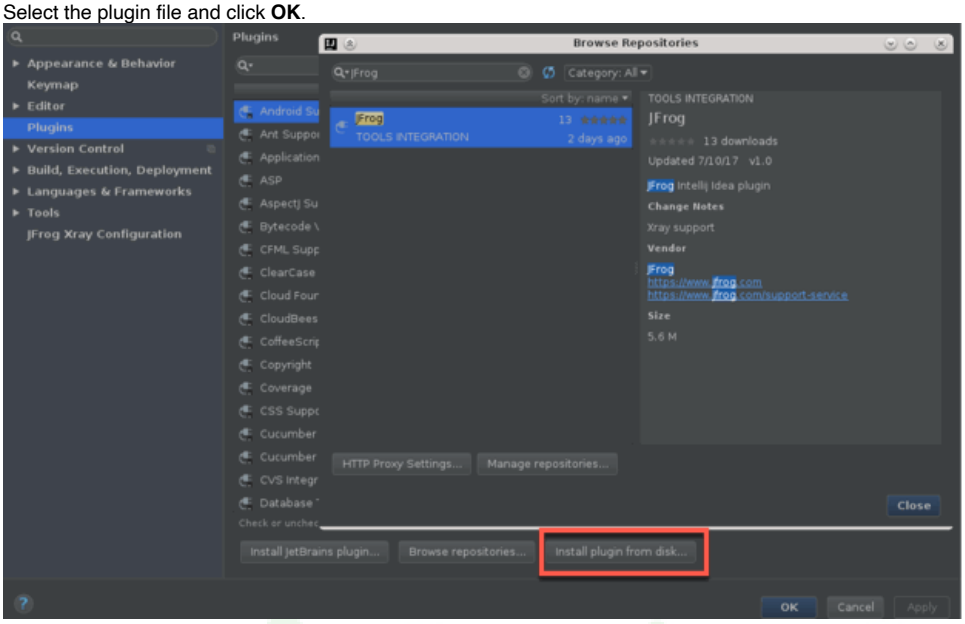
1. Under **Settings (Preferences) | Plugins,** click **Browse repositories** and search for **JFrog**.
2. Once the plugin is found, click **Install JetBrains Plugin**.



Installing Plugin from Disk

1. Download the latest JFrog plugin from **Bintray** or create this plugin from sources. To learn more about building from sources, see the procedure **in GitHub**.
2. Under **Settings (Preferences) | Plugins**, click **Install plugin from disk...**

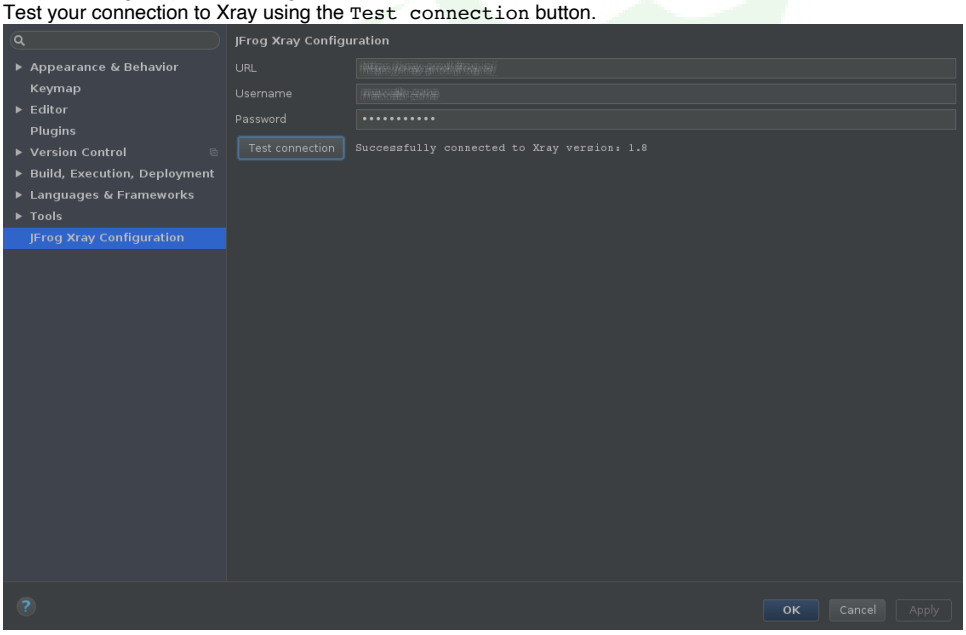
3. Select the plugin file and click **OK**.



Configuring the Plugin to Connect to JFrog Xray

Once the plugin is successfully installed, connect the plugin to your instance of JFrog Xray.

1. Under **Settings (Preferences) | Other Settings**, click **JFrog Xray Configuration**.
2. Set your JFrog Xray URL and login credentials.
3. Test your connection to Xray using the `Test connection` button.



Scanning and Viewing the Results

JFrog Xray automatically performs a scan whenever there is a change in the dependencies in the project.

To manually invoke a scan:

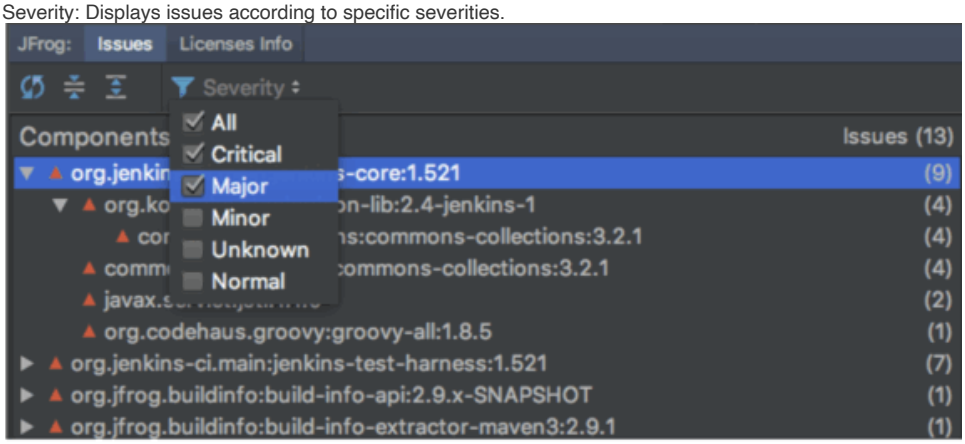
Click Refresh in the JFrog plugin.

View the scanned results in the plugin.

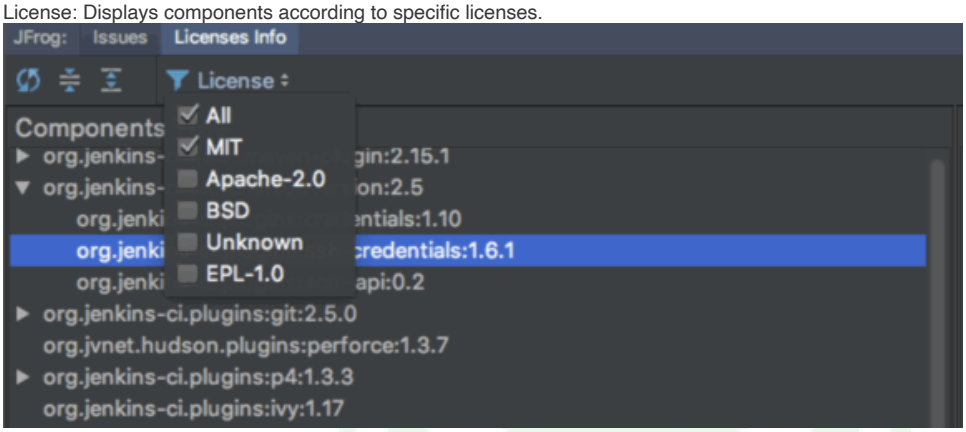
Filtering Xray Scanned Results

The JFrog plugin provides the following filters to narrow down the scanned results to view exactly what you need:

- Severity: Displays issues according to specific severities.



- License: Displays components according to specific licenses.



Watch the Screencast

Watch this screencast to learn how the JFrog IntelliJ IDEA plugin adds JFrog Xray scanning of Maven project dependencies to your IntelliJ IDEA.

System Maintenance and Monitoring

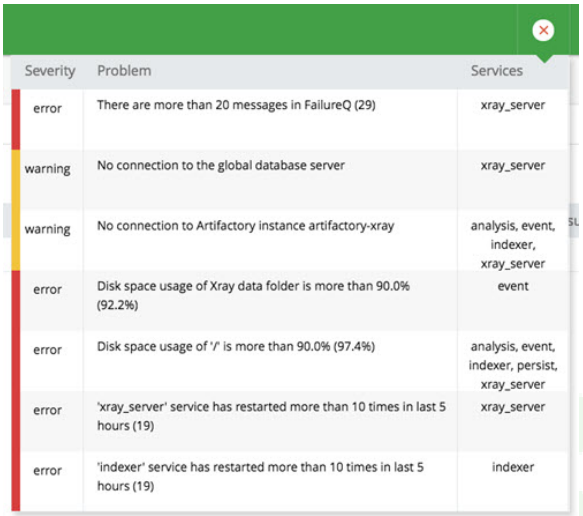
Overview

Xray provides different facilities that allow you to maintain and monitor your installation.

Page contents

System Status

Xray displays your general system status as an icon in the top ribbon. Clicking the icon provides status details.



Severity	Problem	Services
error	There are more than 20 messages in FailureQ (29)	xray_server
warning	No connection to the global database server	xray_server
warning	No connection to Artifactory instance artifactory-xray	analysis, event, indexer, xray_server
error	Disk space usage of Xray data folder is more than 90.0% (92.2%)	event
error	Disk space usage of '/' is more than 90.0% (97.4%)	analysis, event, indexer, persist, xray_server
error	'xray_server' service has restarted more than 10 times in last 5 hours (19)	xray_server
error	'indexer' service has restarted more than 10 times in last 5 hours (19)	indexer

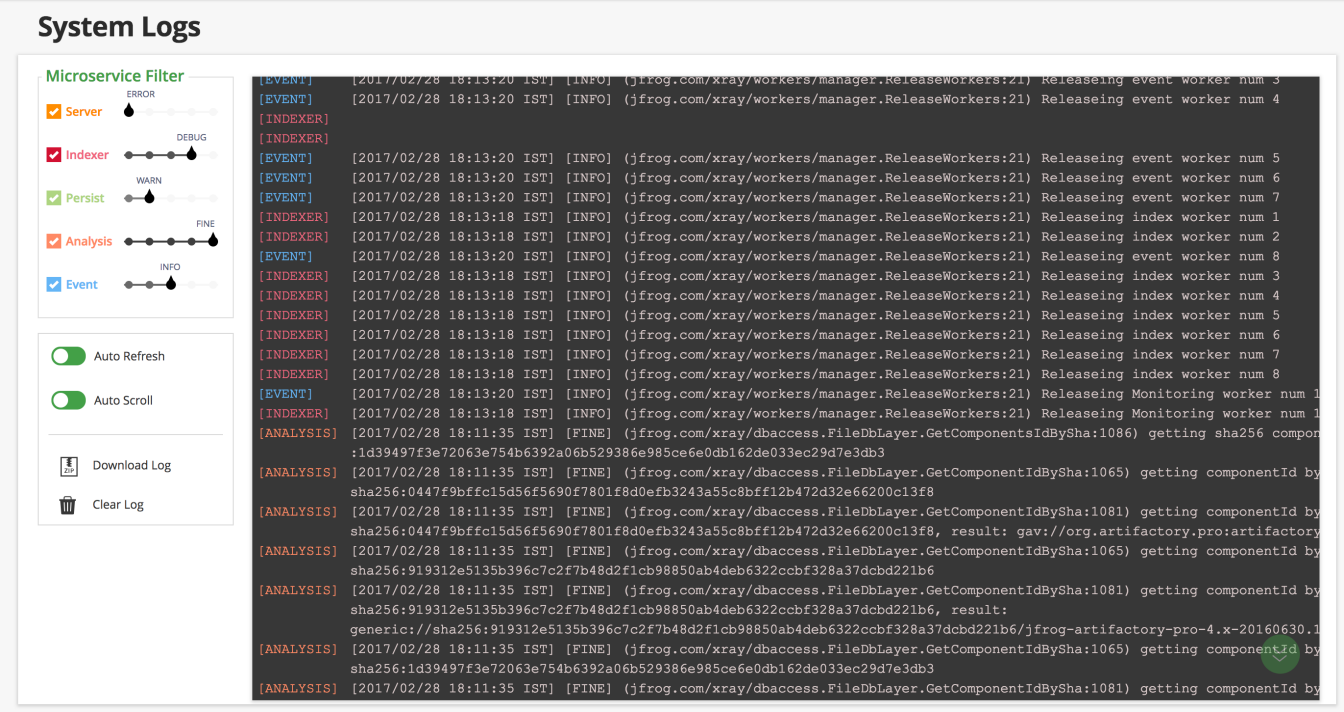
The overall status is an accumulation of the status for a variety of parameters. The table below describes the parameters monitored and the condition that generates a notification. Note that some notifications may be categorized as "warnings" or "errors" depending on the severity of the condition:

Parameter	Condition for notification
Connection to the PostgreSQL database	No connection
Connection to MongoDB database	No connection
Connection to RabbitMQ messaging service	No connection
Connection to Global Database Server	No connection
Connection to Artifactory instances	No connection
Connection to integrated services (e.g. Whitesource)	No connection
Service restarts	**Warning**: 3 in the last 5 hours **Error**: 50 in the last 5 hours
Average CPU usage	**Warning:** 90% **Error:** 95%
Average RAM usage	**Warning:** 90% **Error:** 95%
System open files usage	**Warning:** 80% of maximum **Error:** 95% of maximum
Working directory disk usage	**Warning:** 80% of total **Error:** 95% of total
Xray data folder disk usage	**Warning:** 80% of maximum specified in the Xray configuration files **Error:** 95% of maximum specified in the Xray configuration files
Failed Messages Count	**Warning:** More than 0 failure messages

	Error: More than 100 failure messages

System Logs

You can view Xray's system logs from the **Admin** module under **System Logs.**

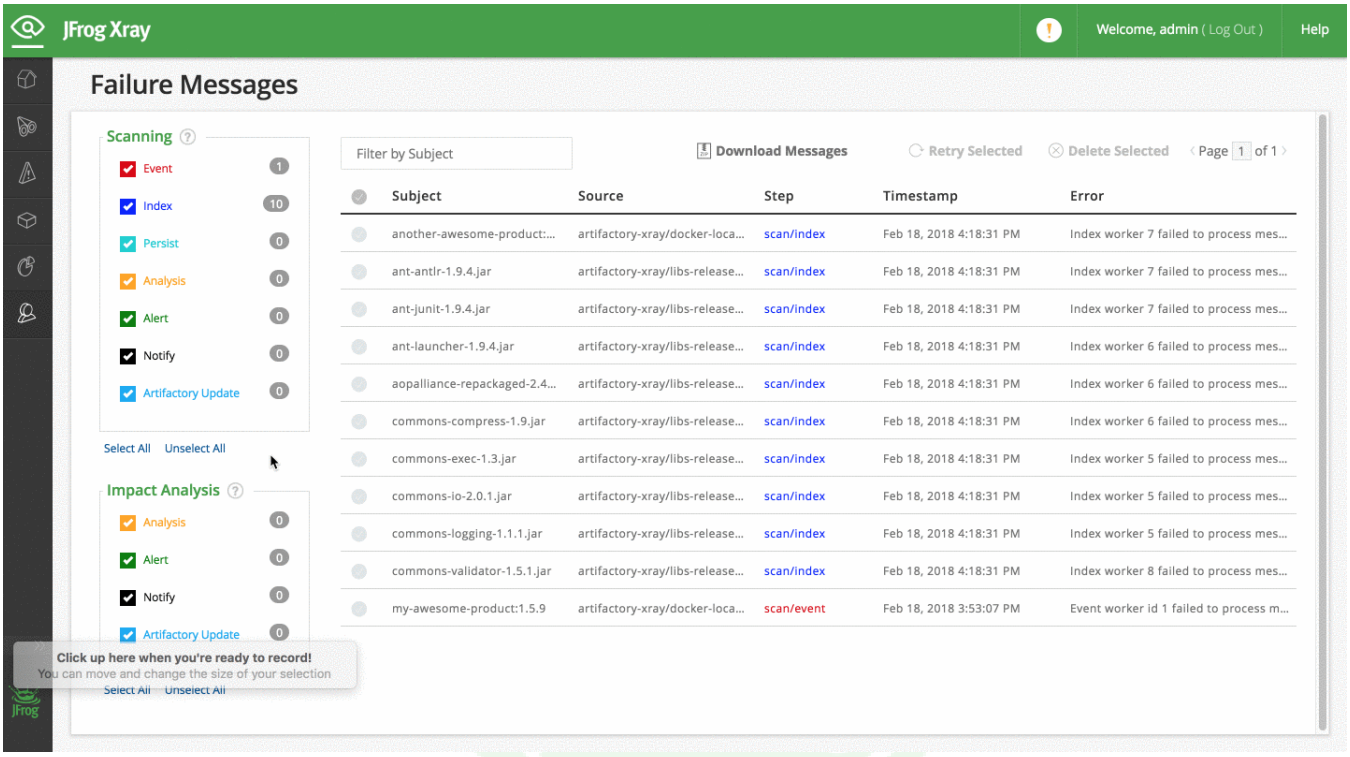


Microservice Filter	Click on any of the microservices listed to enable or disable logging from that microservice. For each microservice you can set the log level to one of ERROR, WARN, INFO, DEBUG or FINE.
Auto Refresh	When enabled, the display will automatically refresh itself as new log entries are added.
Auto Scroll	When enabled, the display will automatically scroll as new log entries are added.
Download Log	Click to download a hard copy of the log file in its current state.
Clear Log	Click to clear the display. This does not remove any entries from the actual log file.

Failure Messages

Xray administrators can view a list of all artifact and data failure messages in the Failure Messages page, under the Admin module. Each failure can be traced to the exact step in the **scanning** and **impact analysis** Xray process in which it failed, allowing administrators to fix the issue and retry the step. Or contact JFrog support for further investigation.

Search for specific failures using the "Filter by Subject" box and by selecting the specific scan and impact steps.



Subject	The name of the failed artifact being scanned by Xray, or the data update in case of an impact analysis, such as vulnerability and licence name.
Source	The location of the artifact being scanned by Xray (including the instance name, repo name and path within the repo), or the source of the data update in case of impact analysis (including the database sync or custom issue assigned).
Step	The step in which the artifact failed (including the process and step name).
Timestamp	The time in which the failure occurred. By default, the grid will be sorted from newest failure to oldest.
Error	The detailed error message describing what caused this failure.

Scanning

Every time a new artifact or build is added to a connected Artifactory instance, Xray scans it and its dependencies for known vulnerabilities and compliance violations and generate Issues accordingly. This process is called "Scanning". That includes the following process steps:

1. Event
2. Index
3. Persist
4. Analysis
5. Alert
6. Notify
7. Artifactory Update

Impact Analysis

Every time new component metadata is available (vulnerabilities, licenses, etc.), Xray looks up the component in the components graph and if the update matches any watches, Xray will generate an issue and create a map of its impact to determine which artifacts are ultimately affected by it. This process is called "Impact Analysis". That includes the following process steps:

1. Analysis
2. Alert
3. Notify
4. Artifactory Update

Backup and Restore

JFrog Xray is made up of several " Go" services as well as some external services. Each needs to be handled separately.

Xray's backup and restore solution is based on storage snapshotting to store data and configuration. In order to ensure data consistency and reliability, all Xray services must flush their data to disk before running the snapshot tool or keep a transaction log.

Backup Directories

The directories that should be backed up depend on whether you are running Xray in a Docker container or not.

Docker Backup Directories

To back up Xray running in a Docker container, you need to back up the $XRAY_MOUNT_ROOT directory which contains the following sub-directories:

- xray
- postgres
- rabbitmq
- mongodb

For example, if $XRAY_MOUNT_ROOT=/root/.jfrog/xray, you would need to back up the following directories:

- /root/.jfrog/xray/xray
- /root/.jfrog/xray/postgres
- /root/.jfrog/xray/rabbitmq
- /root/.jfrog/xray/mongodb

Non-Docker Backup Directories

For non-Docker distributions, Xray's data is distributed among the following directories:

XRay Go services	/var/opt/jfrog/xray/data
PostgreSQL	/var/opt/jfrog/postgres/data
RabbitMQ	For a list of default Linux directories, please refer to the RabbitMQ documentation. The directories relevant for Xray are: ▪ /etc/rabbitmq ▪ /var/lib/rabbitmq/mnesia
MongoDB	/var/lib/mongodb

Running a Backup

To run a backup, simply create a snapshot of each of the backup directories described in the previous section.

Restoring from a Backup

Before restoring from a snapshot, we recommend backing up your current state.

To restore Xray from a backup:

1. Stop Xray
2. Overwrite the data in the backup directories with the corresponding data in your backup
3. Start Xray

Xray REST API

Overview

Xray provides a convenient and up-to-date self-descriptive API that can be used by various tools /frameworks to automate the creation of REST calls.

Usage

Xray REST API endpoints can be invoked in any of the standard ways to invoke a RESTful API. This section describes how to use the Artifactory REST API using cURL as an example.

> **Using and Configuring cURL**
>
> You can download cURL here. Learn how to use and configure cURL here.

Base URL

http://<xrayhost>:<port>/api/v1

Authentication

Most REST API calls need to be authenticated using your Xray user and password or through an authentication token. A few calls (such as SYSTEM calls) do not require authentication.

Example - Deleting a Watch

The example below demonstrates how to invoke the Delete Watch REST API with the following assumptions:

- You are using cURL from the unix command line, and are presently working from the **home (~) directory** of the user '**myUser**':
- You wish to delete the watch called *performance-watch*.
- You have Xray running on your local system, on **port 8000**.
- You have configured a user in Xray named 'myUser', with password 'myP455w0rd!'.
- You have an authentication token with value 12345

To execute a call using basic authentication you would run:

```
curl -u myUser:myP455w0rd! -X DELETE http://localhost:8000/api/v1/watches
/performance-watch
```

To execute a call authenticating with a token you would run:

```
curl X DELETE http://localhost:8000/api/v1/watches/performance-watch?
token=12345
```

Component Identifiers

Several endpoints require the use of a component identifier which must be formatted, according to its package type, using the convention described in the following table:

Package Type	Identifier	Example
Maven	gav://group:artifact:version	gav://ant:ant:1.6.5
Docker	docker://Namespace/name:tag	docker://jfrog/artifactory-oss:latest
RPM	rpm://dist(optional):arch:name:version	rpm://el6:i386:ImageMagick:6.7.2.7-4
Debian	deb://dist(optional):arch:name:version	deb://lucid:i386:acl:2.2.49-2
NuGet	nuget://module:version	nuget://log4net:9.0.1

Page Contents

Generic file	generic://sha256:<Checksum>:name	generic://sha256:244fd47e07d1004f0aed9c156aa09083c82bf8944eceb67c946ff7430510a77b:foo.jar
NPM	npm://package:version	npm://mocha:2.4.5
Python	pip://package:version	pip://raven:5.13.0

REST Resources

USER MANAGEMENT

Create User

Description: Creates a new Xray User.
Security: Requires an admin user
Usage: POST /users
Consumes: application/json

```
{
  "admin": <true | false>,
  "email": "",
  "name": "",
  "password": ""
}
```

Response Codes:
200: Success - User created
400: Cannot create user with suffix _xray
409: User with name {name} already exists
415: Failed to parse request
500: Failed to check if user exists in the database
500: Failed to marshal response

Update User

Description: Updates an Xray User.
Security: Requires an admin user
Usage: PUT /users/{id}
Consumes: application/json

```
{
  "admin": <true | false>,
  "email": "",
  "name": "",
  "password": ""
}
```

Response Codes:
200: Success - User updated
400: Failed to update a new user
400: Mail format is not valid
404: Failed to find user
415: Failed to parse request

Get Users/ Get User

Description: Gets a list of all users in the system or a specific user
Security: Requires an admin user
Usage: GET /users I GET /user/{id}
Produces: application/json

```
[
{
  "admin": <true | false>,
  "email": "",
```

```
  "name": "",
  "password": ""
}
]
```

Response Codes:
200: Success
404: Use with id {id} does not exist
500: Failed to serialize user data
500: Failed to retrieve user
500: Failed to retrieve user {id}

Delete User

Description: Deletes a user
Security: Requires an admin user
Usage: DELETE /users/{id}
Response Codes:
200: Success - user was deleted
403: User cannot delete itself {id}
404: Failed to retrieve user {id}
500: User "admin" cannot be deleted

ISSUES

Create Issue Event

Description: Allows an issue vendor to create a new issue event
Security: Requires a valid user with "Manage Components" permission
Usage: POST /events
Consumes: application/json

```
{
        "type" : "<issue type>",
        "source_id" : "<vendor unique identifier>",
        "url" : "<url for issue information>",
        "created" : "<creation date in ISO8601 format (yyyy-MM-dd'T'HH:mm:
ss.SSSZ)>",
        "description" : "Description of the event",
        "provider" : "The provider of the issue",
        "severity": "",
        "source_id": "",
        "summary": "",
        "updated": "<update time in ISO8601 format (yyyy-MM-dd'T'HH:mm:ss.
SSSZ)>",
        "modified": "<modification time in ISO8601 format (yyyy-MM-dd'T'HH:
mm:ss.SSSZ)>",
        "components" : //A list of components affected with this issue
                [
                        {
                                "component_id" : "<component unique
identifier>",
                                "properties" : {["key" : "value"]}
                        }
                ],
        "properties" : {["key" : "value"]}
}
```

Response: 201
Sample usage:

```
POST /events
{
  "type": "security",
  "source_id": "574eb686ee6061000b7a6429",
  "url": "http://more.info",
  "created": "2016-05-03T07:30:51.991",
  "Components": [
```

```
    {
      "component_id": "gav://org.apache.maven:maven-settings:3.0.4",
      "properties": {
        "performance": false
      }
    }
  ],
  "properties": {
    "cve": "CVE-2012-2098",
    "summary": "Algorithmic complexity issue",
    "description": "Algorithmic complexity issue in...",
    "cvss_v2": "critical"
  }
}
```

Update Issue Event

Description: Allows an issue vendor to update an issue event
Security: Requires a valid user with "Manage Components" permission
Usage: PUT /events
Consumes: application/json

```
{
        "type" : "<issue type>",
        "source_id" : "<vendor unique identifier>",
        "url" : "<url for issue information>",
        "created" : "<creation date in ISO8601 format (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
        "components" : //A list of components affected with this issue
                [
                        {
                                "component_id" : "<component unique identifier>",
                                "properties" : {["key" : "value"]}
                        }
                ],
        "properties" : {["key" : "value"]}
}
```

Response: 201
Sample usage:

```
PUT /events/574eb686ee6061000b7a6429

{
  "type": "security",
  "source_id": "574eb686ee6061000b7a6429",
  "url": "http://more.info",
  "created": "2016-05-03T07:30:51.991",
  "Components": [
    {
      "component_id": "gav://org.apache.maven:maven-settings:3.0.4",
      "properties": {
        "performance": false
      }
    }
  ],
  "properties": {
    "cve": "CVE-2012-2098",
    "summary": "Algorithmic complexity issue",
    "description": "Algorithmic complexity issue in...",
    "cvss_v2": "critical"
  }
}
```

Get Issue Events

Description: Gets an issue created by a vendor
Security: Requires a valid user with "View Components" permission
Usage: GET /events/{sourceId}
Produces: application/json
Response: 201
Sample usage:

```
GET /events/574eb686ee6061000b7a6429

{
        "type": "security",
          "source_id": "574eb686ee6061000b7a6429",
          "url": "http://more.info",
          "created": "2015-11-03T07:30:51.991",
          "Components": [
    {
            "component_id": "gav://org.apache.maven:maven-settings:3.0.4",
            "properties": { "performance": false }
    }
          ],
          "properties": {
                    "cve": "CVE-2012-2098",
            "summary": "Algorithmic complexity issue",
            "description": "Algorithmic complexity issue in...",
            "cvss_v2": "critical"
          }
}
```

WATCHES

Create Watch

Description: Creates a new Watch. Mandatory fields are "name" and "target_type". If "target_type" is not "artifact", then "target_name" and "art_id" are also mandatory.
Security: Requires a valid user with "Manage Watches/Alerts" permission
Usage: POST /watches
Consumes: application/json

```
{
  "active": <true | false>,
  "alerts": 5,
  "art_id": "",
  "description": "",
  "filters": [
    {
      "type": <"regex" | "mime-type" | "build" | "property" | "package-type" | "aql" | "license-black" |
"license-white" | "issue-severity">,
      "value": "interface"
    }
  ],
  "id": "",
  "name": "",
  "post_actions": {
    "emails": [
      ""
    ],
    "slacks": "",
    "webhooks": [
      ""
    ],
        "fail_build":<true|false>
  },
  "repo_type": "",
  "severity": "",
  "system": <true | false>,
  "target_name": "",
```

```
  "target_type": "<"repository" | "build" | "artifact"| "builds">",
  "temp": <true | false>
}
```

Response Codes:
200: Success - Watch created
415: Failed to parse request
400: Watch is not valid. Check mandatory fields
400: Binary manager doesn't exist
409: Watch with name {name} already exists

Update Watch

Description: Updates a Watch. For system watches, only the filters can be updated.
Security: Requires an admin user for a system watch, a valid user with "Manage Watches/Alerts" permission for a user defined watch
Usage: PUT /watches/{name}
Consumes: application/json (Please refer to Create Watch)
Sample usage: (Please refer to Create Watch)
Response Codes:
200: Success. Watch was successfully updated
403: System watch is not editable for non-admin users
404: Failed to update watch. Watch was not found
415: Failed to parse watch
400: Failed to update watch
500: Failed to update watch

Get Watches / Get Watch

Description: Gets a list of all watches in the system or a named watch
Security: Requires a valid user with "View Watches/Alerts" permission
Usage: GET /watches I GET /watches/{name}
Produces: application/json

```
[
{
  "active": <true | false>,
  "alerts": 10,
  "art_id": "",
  "description": "",
  "filters": [
      "type": <"regex" | "mime-type" | "build" | "property" | "package-type" | "aql" | "license-black" |
"license-white" | "issue-severity">,
      "value": "interface"
  ],
  "id": "",
  "name": "",
  "post_actions": {
    "emails": [
      ""
    ],
    "slacks": "",
    "webhooks": [
      ""
    ],
        "fail_build":<true|false>
  },
  "repo_type": "",
  "severity": "",
  "system": <true | false>,
  "target_name": "",
  "target_type": "<"repository" | "build" | "artifact"| "builds">",
  "temp": <true | false>
}
]
```

Sample usage:

```
GET /watches/Apache-2.0%20-%20banned?token=12345

{
  "id": "581f320e40a1632602462bb1",
  "name": "Apache-2.0 - banned",
  "description": "",
  "art_id": "",
  "active": true,
  "post_actions": {
    "emails": [],
    "webhooks": [],
        "fail_build":<true|false>
  },
  "filters": [
    {
      "type": "license-black",
      "value": [
        "Apache-2.0"
      ]
    }
  ],
  "target_type": "artifact",
  "alerts": 0
}
```

Response Codes:
200: Success
500: Failed to get watches

Delete Watch

Description: Deletes a watch
Security: Requires a valid user with "Manage Watches/Alerts" permission
Usage: DELETE /watches/{watch_name}

Produces: application/json

```
{
  "active": <true | false>,
  "alerts": 10,
  "art_id": "",
  "description": "",
  "filters": [
    {
      "type":"",
      "value":""
    }
  ],
  "id": "",
  "name": "",
  "post_actions": {
    "emails": [
      ""
    ],
    "slacks": "",
    "webhooks": [
      ""
    ],
        "fail_build":<true|false>
  },
  "repo_type": "",
  "severity": "",
  "system": <true | false>,
  "target_name": "",
  "target_type": "",
  "temp": <true | false>
}
```

Response Codes:

200: Success - watch(es) deleted

400: Failed to delete watches

ALERTS

> **DEPRECATION NOTICE**
>
> From version 1.12, Alerts have been deprecated and are replaced by the Violations feature.
>
> Ignore Rules can now be viewed the Watches screen. Similarly, and any Alerts present in your system when upgrading to version 1.12 are still available in the Archived Alerts tab of the Watches screen. Ignore Rules and Archived Alerts will be completely removed in forthcoming versions.

Get User Alerts

Description: Gets all current alerts for a user
Security: Requires a valid user with "View Watches/Alerts" permission
Usage: GET /alerts?offset=<num_of_alerts_to_skip>&limit=<max_alerts_to_display>&direction=<asc | desc>&order_by=<created_at | target | top_severity | watch_name>
Produces: application/json

```
[
        {
            "alert_id":"5a3158a9c0d05100014931d4",
        "created_at": "<creation date in ISO8601 format (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
                "top_severity": "<top severity for all issues reported in the alert>",
                "watch_name": "<The name of the watch for which this alert was issued>",
                "issues" : [
              {
                "severity" : "<severity of the issue>",
                        "type" : "<issue type>",
                        "provider" : "<feed provider>",
                        "created": "<creation date in ISO8601 format (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
                        "summary" : "<Summary of the issue>",
                        "component_ids" : ["<component_id>"],
                                "description" : "<More detailed description of the
issue>",
                                "impacted_artifacts" : [
                                {
                                        "name": "<file name>",
                                        "display_name": "<name as displayed in Artifactory>",
                                        "path": "<file path in repository>",
                                        "pkg_type": "<The package type of the artifact for which this alert
was issued>",
                                        "sha256": "<file SHA256 checksum>",
                                        "sha1": "<file SHA1 checksum>",
                                        "depth": "<file depth in impact path>",
                                        "parentSha": "<SHA256 checksum of parent file>",
                            "impact_path:": "<path to the infected file>",
                                   "infected_file": {
                                      "name": "<infected file name>",
                                                "path": "<infected file name>",
                                      "sha256": "<infected file SHA256 checksum>",
                                      "sha1": "<infected file SHA1 checksum>",
                                                "depth": "<infected file depth in impact path>",
                                      "parent_sha": "<parent artifact file SHA256 checksum>",
                                                "display_name": "<infected file name as displayed in
Artifactory>",
                                       "pkg_type": "<infected file package type>"
                                   }
                            }
                        ]
              }
    ]
  }
]
```

Sample usage:

```
GET /alerts?offset=0&limit=25&order_by=created_at&direction=desc

Response:
  "data": [
    {
      "created": "2016-12-07T14:12:08.466Z",
      "top_severity": "Critical",
      "watch_name": "new-watch",
      "issues": [
        {
          "severity": "Major",
          "type": "security",
          "provider": "WhiteSource",
          "created": "0001-01-01T00:00:00Z",
          "summary": "FileSystemBytecodeCache in Jinja2 2.7.2 does not properly create temporary
directories",
          "component_ids": [
            "pypi://Jinja2:2.7.2"
          ],
          "description": "FileSystemBytecodeCache in Jinja2 2.7.2 does not properly create temporary
directories, which allows local users to gain privileges by pre-creating a temporary directory with a user's
uid.  NOTE: this vulnerability exists because of an incomplete fix for CVE-2014-1402.",
          "impacted_artifacts": [
            {
              "name": "Jinja2-2.7.2.tar.gz",
              "display_name": "Jinja2:2.7.2",
              "path": "artifactory-xray/Python/",
              "pkg_type": "Pypi",
              "sha256": "310a35fbccac3af13ebf927297f871ac656b9da1d248b1fe6765affa71b53235",
              "sha1": "",
              "depth": 0,
              "parent_sha": "310a35fbccac3af13ebf927297f871ac656b9da1d248b1fe6765affa71b53235",
              "impact_path": "",
              "infected_file": {
                "name": "Jinja2-2.7.2.tar.gz",
                "path": "###art12/Python/",
                "sha256": "310a35fbccac3af13ebf927297f871ac656b9da1d248b1fe6765affa71b53235",
                "sha1": "",
                "depth": 0,
                "parent_sha": "310a35fbccac3af13ebf927297f871ac656b9da1d248b1fe6765affa71b53235",
                "display_name": "Jinja2:2.7.2",
                "pkg_type": "Pypi"
              }
            }
        ]
      }
    ]
  }
]
```

Response Codes:

200: Success - Alerts found
400: Failed to get alerts. Check pagination query parameters.
404: No alerts found
500: Failed to read alerts

Ack Alert

Description: Acknowledge an existing alert
Security: Requires a valid user with "Manage Watches/Alerts" permission
Usage: POST /alerts/ack
Produces: application/json

```
{
        "alert_Ids" : ["<alertId>", "<alertId>",...]
}
```

Sample usage:

```
POST /alerts/ack

Response
{
        "alert_Ids":["576a80bf8571664b691f0c07"]
}
```

Response Codes:

200: Alert has been acknowledged
404: Acknowledged alert not found
500: Failed to acknowledge alert

SCAN

Scan Artifact

Description: Invokes scanning of an artifact
Security: Requires a valid user with "Manage Components" permission
Usage: POST /scanArtifact
Consumes: application/json

```
{
  "checksum": {
    "md5": "",
    "sha1": "",
    "sha256": ""
  },
  "componentId": "",
  "summary": ""
}
```

Response Codes:

200: Scan of artifact is in progress
415: Failed to parse artifact
500: Failed to write message to the queue

Block Download Setup

Description: Configures download blocking for a repository
Security: Requires an admin user
Usage: POST /blockDownloadSetup
consumes: application/json

```
{
  "artifactoryId": "<The ID of the Artifactory instance>",
  "repoKey": "<The repository key in the specified Artifactory instance>",
  "severity": "<The severity of issues found for which downloads should be blocked>"
}
```

Update Block Download Setup

Description: Updates download blocking configuration for a repository
Security: Requires an admin user
Usage: POST /blockDownloadSetup/{name}?watcher_name=<watcher name>
consumes: application/json

```
{
  "artifactoryId": "",
  "repoKey": "",
  "severity": ""
}
```

Scan Build

Description: Invokes scanning of a build that was uploaded to Artifactory as requested by a CI server
Security: Requires an admin user
Usage: POST /scanBuild
Consumes: application/json
Produces: wild card

```
{
 "artifactoryId":<Artifactory instance id>,
 "buildName":<build name>,
 "buildNumber": <build number>
}
```

Produces: application/json

```
{
  "summary": {
    "fail_build": <true | false>,
    "message": <message with more information regarding the fail/success>,
    "more_details_url": <link to all created Alerts in Xray>,
    "total_alerts": <number of alerts generated from the scan>
  },
  "alerts": [  <alert details>
    {
      "created": <creation time of the Alert>,
      "issues": [ <the issues the Alert includes>
        {
          "created": <creation time of the issue>,
          "cve": "",
          "description": <issue description>,
          "impacted_artifacts": [
            {
              "depth": "int",
              "display_name": "",
              "infected_files": [
                {
                  "component_id": "",
                  "depth": "int",
                  "details": [
                    {
                      "banned_licenses": [
                        {
                          "alert_type": "",
                          "description": "",
                          "id": {},
                          "severity": "",
                          "summary": ""
                        }
                      ],
                      "child": "ImpactedFile",
                      "vulnerabilities": [
                        {
                          "alert_type": "",
                          "description": "",
                          "id": {},
                          "severity": "",
                          "summary": ""
                        }
                      ]
                    }
                  ],
```

```
                "display_name": "",
                "name": "",
                "parent_sha": "",
                "path": "",
                "pkg_type": "",
                "sha1": "",
                "sha256": ""
              }
            ],
            "name": "",
            "parent_sha": "",
            "path": "",
            "pkg_type": "",
            "sha1": "",
            "sha256": ""
          }
        ],
        "provider": <issue provider>,
        "severity": <issue severity>,
        "summary": <issue summary>,
        "type": <issue type>
      }
    ],
    "top_severity": <Alert's top severity>,
    "watch_name": <name of the Watch which caused the Alert>
  }
],
"licenses": [
  {
    "name": <license name>
    "components": [<names of build components with this license>],
    "full_name": <license full name>,
    "more_info_url": [<links to more information about this license>],
  }
]
}
```

Response Codes:
200: Build scanned
415: Failed to parse scan build request
400: Request is missing mandatory fields
403: No valid license was found
500: Failed to get Artifactory instance data
500: Failed to check watches
500: Failed to send build to scan

AUTHORIZATION

Get Token

Description: Generates a temporary token that is valid for 2 hours
Security: Requires a valid user
Usage: POST /auth/token
Consumes: application/json

```
{
  "name": "",
  "password": ""
}
```

Sample usage:

```
POST /auth/token

Response
{
  "admin": true,
  "email": "admin@mycompany.com",
```

```
  "token": "12345",
  "userName": "admin"
}
```

Response Codes:

200: Success
415: Bad credentials
500: Failed to marshal response

BINARY MANAGERS

Create Binary Manager Configuration

Description: Configures a new connection to an Artifactory instance
Security: Requires an admin user
Usage: POST /binMgr
Consumes: application/json

```
{
  "binMgrDesc": "",
  "binMgrId": "",
  "binMgrUrl": "",
  "password": "",
  "proxy_enabled": <true | false>,
  "user": ""
}
```

Response Codes:

200: Artifactory instance has been successfully added
400: Failed to create new instance of Artifactory, id is missing
400: Artifactory url cannot contain localhost
401: Bad Credentials
409: Artifactory already exists
409: Artifactory with Id {id} or url {url} already exists
415: Failed to parse request
415: Artifactory id must not contain /
500: Failed to obtain a response
500: Failed to set Artifactory Xray support information
500: Failed to upload Xray compatibility plugin
500: Incompatible version: {version}. This Xray version only supports integration with Artifactory {supportVersion} and above.
500: Failed to check Artifactory configuration
500: Artifactory instances older than version 4.11 must the Enterprise with a valid license
500: Failed to add Artifactory instance to database
500: Failed to create Xray configuration for Artifactory
500: Failed to validate Artifactory license

Get Binary Manager Configurations

Description: Gets the details of all connected Artifactory instances
Security: Requires a valid user
Usage: GET /binMgr
Sample usage:

```
GET /binMgr

[
  {
    "binMgrDesc": "",
    "binMgrId": "",
    "binMgrUrl": "",
    "id": "",
    "license_expired": false,
    "license_valid": false,
    "proxy_enabled": false,
    "version": ""
  }
]
```

Response codes:

200: List of Artifactory instances
401: Bad Credentials
500: Failed to obtain response

Update Backward Compatibility Plugin

Description: Updates the Xray backward compatibility plugin to support versions of Artifactory older than 4.11
Security: Requires an admin user
Usage: POST /updateBinMgrPlugin
Response codes:
200: Xray compatibility plugin has been successfully installed
500: Failed to update Xray compatibility plugin

Update Binary Manager

Description: Updates the details of a connected Artifactory instance
Security: Requires an admin user
Usage: PUT /binMgr/{id}
Consumes: application/json

```
{
  "binMgrDesc": "",
  "binMgrId": "",
  "binMgrUrl": "",
  "password": "",
  "proxy_enabled": <true | false>,
  "user": ""
}
```

Sample usage:

```
PUT /binMgr/Art2
{
  "binMgrDesc": "For QA use",
  "binMgrId": "Art2",
  "binMgrUrl": "http://localhost:8081/artifactory",
  "password": "password",
  "proxy_enabled": false,
  "user": "admin"
}
```

Response Codes:

200: BinMgr has been successfully updated
400: Path parameter is missing
400: Artifactory URL cannot contain localhost
401: Bad Credentials
500: Failed to obtain response

Get Binary Manager

Description: Gets the details of the specified connected Artifactory instance
Security: Requires a valid user
Usage: GET /binMgr/{id}
Sample usage:

```
GET /binMgr/###art12
{
  "binMgrUrl": "http://localhost:8081/artifactory",
  "binMgrId": "###art12",
```

```
  "binMgrDesc": "",
  "version": "4.x-SNAPSHOT",
  "proxy_enabled": false
}
```

Response Codes:

200: Artifactory model
400: Path parameter is missing
401: Bad Credentials
500: Failed to obtain response

Delete Binary Manager

Description: Deletes an Artifactory instance configuration
Security: Requires an admin user
Usage: POST /binMgr/delete
Consumes: application/json

```
{
  "binMgrDesc": "",
  "binMgrId": "",
  "binMgrUrl": "",
  "password": "",
  "proxy_enabled": <true | false>,
  "user": ""
}
```

Sample usage:

```
POST /binMgr/{id}

{
  "binMgrDesc": "For QA",
  "binMgrId": "arti2",
  "binMgrUrl": "http://localhost:8081/artifactory",
  "password": "password",
  "proxy_enabled": true,
  "user": "admin"
}
```

Response Codes:

200: BinMgr has been successfully updated
400: Path parameter is missing
400: Artifactory URL cannot contain localhost
401: Bad Credentials
500: Failed to obtain response

COMPONENTS

Add New Components

Description: Adds new components
Security: Requires an admin user
Usage: POST /component

Consumes: application/json

```
{
  "components": [
    {
      "component": "",
      "created": "2006-01-02T15:04:05Z",
```

```
      "description": "",
      "downloads": 12,
      "licenses": [
        ""
      ],
      "modified": "2006-01-02T15:04:05Z",
      "name": "",
      "package_type": "",
      "sources": [
        {
          "name": "",
          "updated": "2006-01-02T15:04:05Z",
          "url": ""
        }
      ],
      "vcs_url": "",
      "versions": [
        {
          "downloads": 12,
          "files": [
            {
              "md5": "",
              "name": "",
              "sha1": "",
              "sha256": ""
            }
          ],
          "licenses": [
            ""
          ],
          "released": "2006-01-02T15:04:05Z",
          "version": ""
        }
      ],
      "website_url": ""
    }
  ]
}
```


Response Codes:

200: Shows details of an added component

400: Failed to obtain component

500: Failed to add component

Find Component by Name

Description: Search for a component by name - applicable only for components synced from the JFrog Global database to Xray
Security: Requires a valid user with "View Component" permission
Usage: GET /component/{name}
Produces: application/json
Sample usage:


```
GET /component/Jinja2

{
  "component": "Jinja2",
  "package_type": "pip",
  "name": "Jinja2",
  "description": "A small but fast and easy to use stand-alone template engine written in pure python.",
  "website_url": "http://jinja.pocoo.org/",
  "downloads": 35536045,
  "created": "2008-06-09T16:50:19Z",
  "modified": "2016-12-30T06:03:21.705Z",
  "sources": [
    {
```

```
      "name": "pypi",
      "url": "https://pypi.python.org",
      "updated": "2016-12-30T06:03:21.678Z"
    }
  ],
  "versions": [
    {
      "version": "2.8.1",
      "released": "2016-12-29T13:16:20Z",
      "licenses": [
        "BSD"
      ],
      "files": [
        {
          "name": "Jinja2-2.8.1-py2.py3-none-any.whl",
          "sha256": "3997cf273f1424207c60d5895264f74483fce72702f15a7cd51a8551d43663ca",
          "sha1": "805e865181e6bce2f2f6f74f7b54bd913fc54b27",
          "md5": "7472b9df828747c2d44eb539558bbf7a"
        },
        {
          "name": "Jinja2-2.8.1.tar.gz",
          "sha256": "35341f3a97b46327b3ef1eb624aadea87a535b8f50863036e085e7c426ac5891",
          "sha1": "6baee2df662bf193bb3669c2c5b475da6083e2aa",
          "md5": "150a8f1c180272753cf46dd3cdd6decf"
        }
      ]
    }
  ]
}
```


Response Codes:

200: Component found

400: Failed to resolve component mapping

500: Failed to get component by name

Get Artifact Dependency Graph

Description: Get the complete dependency graph for an artifact
Security: Requires a valid user with "View Components" permission
Usage: POST /dependencyGraph/artifact
Consumes: application/json

```
{
  "path": "<artifactory-name/repo-name/path>"
}
```

Produces: application/json

```
{
  "artifact":{
    "name": "<The name of the artifact who's graph we are obtaining>",
    "path": "<artifactory-name/repo-name/path>",
    "pkg_type": "<Package type>",
    "sha256": "<Artifact's SHA256 checksum>",
    "sha1": "<Artifact's SHA1 checksum>",
    "component_id": "<The component ID>"
  },
  "components":[
    {
      "component_name":"<Dependency component name>",
      "component_id":"<Dependency Component ID>",
      "package_type":"<Dependency component package type>",
      "version":"<Dependency component version>",
      "created":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
```

```
      "modified":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
      "components":[<Next level dependencies of the dependency component>]
    }]
}
```

Sample Usage:

```
POST /dependencyGraph/artifact
{
  "path": "/Artifactory/pnnl/goss/goss-core-client/0.1.7/goss-core-client-0.1.7-sources.jar"
}

{
  "artifact":{
    "name": "artifactory-pro.zip",
    "path": "art2/ext-release-local/",
    "pkg_type": "Generic",
    "sha256": "d160c68ed8879ae42756e159daec1dd7ecfd53b6192321656b72715e20d46dd2",
    "sha1": "",
    "component_id": "gav://org.artifactory.pro:artifactory-pro-war:4.14.0"
  },
  "components":[
    {
      "component_name":"some-component-1.1",
      "component_id":"pip://some-component:1.1",
      "package_type":"pip",
      "version":"1.1",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z",
      "components":[]
    },
    {
      "component_name":"some-component-1.2",
      "component_id":"pip://some-component:1.2",
      "package_type":"pip",
      "version":"1.2",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z",
      "components":[
        {
          "component_name":"Jinja2.7.2",
          "component_id":"pip://Jinja2:2.7.2",
          "package_type":"pip",
          "version":"2.7.2",
          "created":"2008-06-09T16:50:19Z",
          "modified":"2015-07-26T17:49:47Z",
          "components":[]
        }
      ]
    }
  ]
}
```

Response Codes:
200: Success
400: Artifact '<PATH>' doesn't exist or isn't indexed in Xray
401: Bad credentials
415: Failed to parse request

Compare Artifacts

Description: Compares two artifacts and produces the difference between them
Security: Requires a valid user with "View Components" permission
Usage: POST /dependencyGraph/artifactDelta
Consumes: application/json

```
{
 "source_artifact_path":"<artifactory/repo/path>",
```

```
 "target_artifact_path":"<artifactory/repo/path>"
}
```

Produces:

```
{
  "source_artifact":{
    "name": "<The name of the source artifact we are comparing>",
    "path": "<artifactory-name/repo-name/path>",
    "pkg_type": "<Package type>",
    "sha256": "<Artifact's SHA256 checksum>",
    "sha1": "<Artifact's SHA1 checksum>",
  },
  "target_artifact":{
    "name": "<The name of the target artifact we are comparing>",
    "path": "<artifactory-name/repo-name/path>",
    "pkg_type": "<Package type>",
    "sha256": "<Artifact's SHA256 checksum>",
    "sha1": "<Artifact's SHA1 checksum>",
  },
  "removed":[
    {
      "component_name":"<Component name only found in source artifact>",
      "component_id":"<Dependency Component ID only found in source artifact>",
      "package_type":"<Dependency component package type>",
      "version":"<Dependency component version>",
      "created":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
      "modified":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>"
    }
  ],
  "added":[
    {
      "component_name":"<Component name only found in target artifact>",
      "component_id":"<Dependency Component ID only found in target artifact>",
      "package_type":"<Dependency component package type>",
      "version":"<Dependency component version>",
      "created":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
      "modified":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
    }
  ],
 "unchanged":[
    {
      "component_name":"<Component name only found in both artifacts>",
      "component_id":"<Dependency Component ID only found in both artifacts>",
      "package_type":"<Dependency component package type>",
      "version":"<Dependency component version>",
      "created":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
      "modified":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
    }
  ]
}
```

Sample Usage:

```
POST /dependencyGraph/artifactDelta
{
 "source_artifact_path":"/pnnl/goss/goss-core-client/0.1.7/goss-core-client-0.1.7-sources.jar",
 "target_artifact_path":"/pnnl/goss/goss-core-client/0.1.8/goss-core-client-0.1.8-sources.jar",
}

{
  "source_artifact":{
    "name": "artifactory-pro.zip",
    "path": "art2/ext-release-local/",
    "pkg_type": "Generic",
    "sha256": "d160c68ed8879ae42756e159daec1dd7ecfd53b6192321656b72715e20d46dd2",
    "sha1": ""
  },
```

```
  "target_artifact":{
    "name": "artifactory-pro.zip",
    "path": "art2/ext-release-local/",
    "pkg_type": "Generic",
    "sha256": "d160c68ed8879ae42756e159daec1dd7ecfd53b6192321656b72715e20d46dd2",
    "sha1": ""
  },
  "removed":[
    {
      "component_name":"some-component-1.1",
      "component_id":"pip://some-component:1.1",
      "package_type":"pip",
      "version":"1.1",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z"
    }
  ],
  "added":[
    {
      "component_name":"Jinja2.7.2",
      "component_id":"pip://Jinja2:2.7.2",
      "package_type":"pip",
      "version":"2.7.2",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z"
    }
  ],
 "unchanged":[
    {
      "component_name":"Apache1.4",
      "component_id":"gav://apache:1.4",
      "package_type":"maven",
      "version":"1.4",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z"
    }
  ]
}
```

Response Codes:
200: Success
400: Artifact '<PATH>' doesn't exist or isn't indexed in Xray
401: Bad Credentials
415: Failed to parse request

Get Build Dependency Graph

Description: Get the complete dependency graph for a build
Security: Requires a valid user with "View Components" permission
Usage: POST /dependencyGraph/build
Consumes: application/json

```
{
 "artifactory_id":"<Artifactory instance name>",
 "build_name":"<Build name>",
 "build_number":"<Build number>"
}
```

Produces: application/json

```
{
  "build":{
    "name": "<The name of the build who's graph we are obtaining>",
    "path": "<artifactory-name/repo-name/path>",
    "pkg_type": "<Package type>",
    "sha256": "<Artifact's SHA256 checksum>",
    "component_id": "<The component ID>"
  },
```

```
  "components":[
    {
      "component_name":"<Dependency component name>",
      "component_id":"<Dependency Component ID>",
      "package_type":"<Dependency component package type>",
      "version":"<Dependency component version>",
      "created":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
      "modified":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
      "components":[]
    }]
}
```

Sample Usage:

```
POST /dependencyGraph/build
{
 "artifactory_instance":"myInstance",
 "build_name":"someBuild",
 "build_number":"someNumber"
}

{
  "build": {
      "name": "my-build",
      "path": "art2/ext-release-local/",
      "pkg_type": "Generic",
      "sha256": "d160c68ed8879ae42756e159daec1dd7ecfd53b6192321656b72715e20d46dd2",
      "component_id": "gav://org.artifactory.pro:artifactory-pro-war:4.14.0"
    },
  "components":[
    {
      "component_name":"some-component-1.1",
      "component_id":"pip://some-component:1.1",
      "package_type":"pip",
      "version":"1.1",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z",
      "components":[]
    },
    {
      "component_name":"some-component-1.2",
      "component_id":"pip://some-component:1.2",
      "package_type":"pip",
      "version":"1.2",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z",
      "components":[
        {
          "component_name":"Jinja2.7.2",
          "component_id":"pip://Jinja2:2.7.2",
          "package_type":"pip",
          "version":"2.7.2",
          "created":"2008-06-09T16:50:19Z",
          "modified":"2015-07-26T17:49:47Z",
          "components":[]
        }
      ]
    }
  ]
}
```

Response Codes:
200: Success
400: Build '<PATH>' doesn't exist or isn't indexed in Xray
400: Missing build name
400: Missing build number
400: Missing Artifactory ID
401: Bad credentials
415: Failed to parse request

Compare Builds

Description: Compares two builds and produces the difference between them
Security: Requires a valid user with "View Components" permission
Usage: POST /dependencyGraph/buildDelta
Consumes: application/json

```
{
 "source_artifactory_id":"<First instance name>",
 "source_build_name":"<First build name>",
 "source_build_number":"<First build number>",
 "target_artifactory_id":"<Second instance name>",
 "target_build_name":"<Second build name>",
 "target_build_number":"<Second build number>"
}
```

Produces: application/json

```
{
  "source_build":{
    "name": "<The name of the source build we are comparing>",
    "path": "<artifactory-name/repo-name/path>",
    "pkg_type": "<Package type>",
    "sha256": "<Build's SHA256 checksum>",
    "component_id": "<Build's component ID>",
  },
  "target_build":{
    "name": "<The name of the target build we are comparing>",
    "path": "<artifactory-name/repo-name/path>",
    "pkg_type": "<Package type>",
    "sha256": "<Build's SHA256 checksum>",
    "component_id": "<Build's component ID>",
  },
  "removed":[
    {
      "component_name":"<Component name only found in source build>",
      "component_id":"<Dependency Component ID only found in source build>",
      "package_type":"<Dependency component package type>",
      "version":"<Dependency component version>",
      "created":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
      "modified":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>"
    }
  ],
  "added":[
    {
      "component_name":"<Component name only found in target build>",
      "component_id":"<Dependency Component ID only found in target build>",
      "package_type":"<Dependency component package type>",
      "version":"<Dependency component version>",
      "created":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
      "modified":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
    }
  ],
 "unchanged":[
    {
      "component_name":"<Component name only found in both builds>",
      "component_id":"<Dependency Component ID only found in both builds>",
      "package_type":"<Dependency component package type>",
      "version":"<Dependency component version>",
      "created":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
      "modified":"<ISO8601 (yyyy-MM-dd'T'HH:mm:ss.SSSZ)>",
    }
  ]
}
```

Sample Usage:

```
POST /dependencyGraph/buildDelta
{
 "origin_build_artifactory_instance":"my-instance",
 "origin_build_name":"someOriginBuild",
 "origin_build_number":"111",
 "target_build_artifactory_instance":"my-instance",
 "target_build_name":"someTargetBuild",
 "target_build_number":"222",
}

{
  "source_build":{
    "name": "my-build",
      "path": "art2/ext-release-local/",
      "pkg_type": "Generic",
      "sha256": "d160c68ed8879ae42756e159daec1dd7ecfd53b6192321656b72715e20d46dd2",
      "component_id": "gav://org.artifactory.pro:artifactory-pro-war:4.14.0"
  },
  "target_build":{
    "name": "my-build",
      "path": "art2/ext-release-local/",
      "pkg_type": "Generic",
      "sha256": "d160c68ed8879ae42756e159daec1dd7ecfd53b6192321656b72715e20d46dd2",
      "component_id": "gav://org.artifactory.pro:artifactory-pro-war:4.14.0"
  },
  " removed":[
    {
      "component_name":"some-component-1.1",
      "component_id":"pip://some-component:1.1",
      "package_type":"pip",
      "version":"1.1",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z"
    }
  ],
  "added":[
    {
      "component_name":"Jinja2.7.2",
      "component_id":"pip://Jinja2:2.7.2",
      "package_type":"pip",
      "version":"2.7.2",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z"
    }
  ],
 "unchanged":[
    {
      "component_name":"Apache1.4",
      "component_id":"gav://apache:1.4",
      "package_type":"maven",
      "version":"1.4",
      "created":"2008-06-09T16:50:19Z",
      "modified":"2015-07-26T17:49:47Z"
    }
  ]
}
```

Response Codes:
200: Success
400: The build with the provided identifier doesn't exist or isn't indexed in Xray
401: Bad credentials
415: Failed to parse request

VULNERABILITIES

Create a New Issue Event

Description: Enables an Issue Vendor to Create a New Issue Event in Xray
Security: Requires a valid user

Usage: POST /events
Consumes: application/json

```
{
  "components": [
    {
      "component_id": "",
      "properties": [
        ""
      ]
    }
  ],
  "created": "2006-01-02T15:04:05Z",
  "description": "",
  "id": "",
  "modified": "2006-01-02T15:04:05Z",
  "properties": [
    "interface"
  ],
  "provider": "",
  "severity": "",
  "source_id": "",
  "summary": "",
  "type": "",
  "updated": "2006-01-02T15:04:05Z",
  "url": ""
}
```

Response codes:

200: Vulnerability data added successfully

500: Failed to add vulnerability data

Gets Single Issue Event

Description: Retrieves one vulnerability issue created by a vendor
Security: Requires a valid user
Usage: GET /events/{componentId}
Produces: application/json

Response codes:

200: Vulnerability issue retrieved successfully

404: Failed to get vulnerability issue

500: Vulnerability issue not found

Allows an issue vendor to update an issue event

Description: Enables an Issue Vendor to Create a New Issue Event in Xray
Security: Requires a valid user
Usage: PUT /events

Consumes: application/json

```
{
  "components": [
    {
      "component_id": "",
      "properties": [
        ""
      ]
    }
  ],
  "created": "2006-01-02T15:04:05Z",
  "description": "",
```

```
  "id": "",
  "modified": "2006-01-02T15:04:05Z",
  "properties": [
    "interface"
  ],
  "provider": "",
  "severity": "",
  "source_id": "",
  "summary": "",
  "type": "",
  "updated": "2006-01-02T15:04:05Z",
  "url": ""
}
```

Response Codes:

200: Vulnerability has been updated successfully

500: Failed to update vulnerability

INDEX

Index artifact

Description: Indexes an artifact
Security: Requires an admin user
Usage: POST /index
Consumes: application/json

```
{
 "archivePath": "",
 "artifactoryId": "",
 "buildName": "",
 "buildNumber": "",
 "checksums": {
   "md5": "",
   "sha1": "",
   "sha256": ""
 },
 "componentId": "",
 "created": 123,
 "downloadUrl": "",
 "downloadedArchive": [
   {
     "ArchiveName": "",
     "DownloadLink": "",
     "SavedFilePath": "",
     "Sha": ""
   }
 ],
 "eventTime": 123,
 "eventType": "",
 "existArchive": [
   {
     "ArchiveName": "",
     "DownloadLink": "",
     "SavedFilePath": "",
     "Sha": ""
   }
 ],
 "messageId": "",
 "origEventType": "",
 "path": "",
 "repoKey": "",
```

```
  "sourcePath": "",
  "sourceRepoKey": ""
}
```

Response Codes:

200: Index event successfully sent to queue

415: Unsupported media type

500: Failed to update event states for message

INTEGRATION

Get Integration Configuration

Description: Retrieves integrations configured into the system
Security: Requires an admin user
Usage: GET /integration

Produces: application/json

Sample usage:

```
GET /integration

 [
  {
    "vendor": "whitesource",
    "api_key": "4a547ccd-fdf0-4ac4-8ec2-259ce91c1633",
    "enabled": <true|false>,
    "context": "project_id",
    "url": "https://saas.whitesourcesoftware.com/xray",
    "description": "WhiteSource provides a simple yet powerful open source security and licenses management
solution. More details at http://www.whitesourcesoftware.com.",
    "test_url": "https://saas.whitesourcesoftware.com/xray/api/checkauth"
  }
]
```

Response Codes:

200: Integration data retrieved successfully

500: Failed to retrieve integration data

Add Integration Configuration

Description: Add an integration configuration
Security: Requires an admin user
Usage: POST /integration

Consumes: application/json

```
 {
  "vendor": ""
  "api_key": "",
  "enabled": <true|false>,
  "context": "",
  "url": "",
  "description": "",
  "test_url": ""
}
```

Sample usage:

```
POST /integration

{
  "vendor": "whitesource",
  "api_key": "12345",
  "enabled": true,
  "context": "project_id",
  "url": "https://saas.whitesourcesoftware.com/xray",
  "description": "WhiteSource provides a simple yet powerful open source security and licenses management
solution. More details at http://www.whitesourcesoftware.com.",
  "test_url": "https://saas.whitesourcesoftware.com/xray/api/checkauth"
}
```

Response Codes:

200: Integration data successfully added

500: Failed to register integration data

Update Integration Configuration

Description: Updates the integration configuration
Security: Requires an admin user
Usage: PUT /integration/{name}
Consumes: application/json

```
 {
  "vendor": "",
  "api_key": "",
  "enabled": <true|false>,
  "context": "",
  "url": "",
  "description": "",
  "test_url": ""
}
```

Response Codes:

200: Integration data successfully Updated

500: Failed to register integration data

Delete Integration Configuration

Description: Delete integration configuration
Security: Requires an admin user
Usage: DELETE /integration/{name}
Produces: application/json

Sample usage:

```
 DELETE /integration/whitesource
```

Response Codes:

200: Integration deleted successfully

400: Vendor name is missing

500: Failed to delete integration

MONITORING

Get System Monitoring Status

Description: Gets system monitoring status
Security: Requires a valid user
Usage: GET /monitor
Produces: application/json

Sample usage:

```
GET /monitor

{
 "problems": [
   {
     "severity": "warning",
     "services": [
       "analysis",
       "event",
       "indexer",
       "xray_server"
     ],
     "problem": "No connection to Artifactory instance ###art12"
   }
 ]
}
```

Response Codes:

200: System monitoring status was sent

500: Failed to marshal object to json

LICENSE REPORTS

Get License Report

Description: Gets a report describing the distribution and compliance status of licenses in all indexed components
Security: Requires a valid user with "Generate Reports" permission
Usage: GET /licensesReport
Produces: application/json

Sample usage:

```
GET /licensesReport
{
  "distribution": {
    "Apache-2.0": 24,
    "MIT": 10,
    "Other": 1,
    "Unknown": 333
  },
  "compliance": {
    "banned": 25,
    "unknown": 333,
    "valid": 10
  },
  "lastUpdate": "2017-01-01T12:20:49.024397147Z"
}
```

Response Codes:

200: Successfully obtained license report

Get License Report Components

Description: Get license report. either `license` or `compliance` query parameter are required
Security: Requires a valid user with "Generate Reports" permission
Usage: GET /licensesReport/components?[license=<license(BSD, GPL etc.)> | compliance=<unknown | valid | banned>]&direction=<asc | desc>&num_of_rows=<rows>&order_by=<id | licenses>&page_num=<page to view>
Produces: application/json
Sample usage:

```
GET /licensesReport/components?
compliance=valid&token=12345&direction=asc&num_of_rows=20&order_by=id&page_num=1

{
  "data": [
    {
      "component_id": "npm://backbone.datastore:1.0.3",
      "component_name": "backbone.datastore:1.0.3",
      "pkg_type": "Npm",
      "is_root": false,
      "licenses": [
        "MIT"
      ],
      "watches": [
        "MIT-allowed"
      ]
    },
        ...
}
```

Response Codes:

200: Successfully obtained license report components

400: Failed to get report components - check pagination query parameters.

500: Failed to obtain report components

Run License Report Generation

Description: Generate synchronized licensed report and wait for results
Security: Requires a valid user with "Generate Reports" permission
Usage: GET /licensesReport/generate
Produces: application/json

Sample usage:

```
{
  "info": "License report was regenerated"
}
```

Response Codes:

200: License report was regenerated

500: Failed to regenerate license report

SECURITY REPORTS

Generate Synchronized Security Report

Description: Generate synchronized security report and wait for results
Security: Requires a valid user with "Generate Reports" permission
Usage: GET /securityReport/generate
Produces: application/json

```
{
  "info": "security report was regenerated"
}
```

Response Codes:

200: Security report was regenerated

500: Failed to regenerate security report

Get Security Report

Description: Get the security report
Security: Requires a valid user with "Generate Reports" permission
Usage: GET /securityReport
Produces: application/json

```
{
  "lastUpdate": "Time",
  "recent_components": [
    <int>
  ],
  "recent_vulnerabilities": [
    <int>
  ],
  "top_artifacts": [
    {
      "indexed": "Time",
      "name": "",
      "package_type": "",
      "path": "",
      "vulnerabilities": [
        {
          "id": {},
          "summary": ""
        }
      ]
    }
  ],
  "top_vulnerabilities": [
    {
      "affected_components": [
        {
          "id": "",
          "impacted_paths": [
            ""
          ]
        }
      ],
      "created": "Time",
      "description": "",
      "properties": [
        "interface"
      ],
      "severity": "",
```

```
      "summary": ""
    }
  ]
}
```

Sample usage:

```
GET /securityReport

{
  "recent_vulnerabilities": {
    "2016-12-04": 0,
    "2016-12-11": 0,
    "2016-12-18": 0,
    "2016-12-25": 0,
    "2017-01-01": 3
  },
  "recent_components": {
    "2016-12-04": 0,
    "2016-12-11": 0,
    "2016-12-18": 0,
    "2016-12-25": 46,
    "2017-01-01": 0
  },
  "top_vulnerabilities": [
    {
      "summary": "CWE-20 Improper Input Validation",
      "description": "The MultipartStream class in Apache Commons Fileupload before 1.3.2, as used in Apache
Tomcat 7.x before 7.0.70, 8.x before 8.0.36, 8.5.x before 8.5.3, and 9.x before 9.0.0.M7 and other products,
allows remote attackers to cause a denial of service (CPU consumption) via a long boundary string.",
      "severity": "Critical",
      "properties": {
        "cve": "CVE-2016-3092",
        "cvss_v2": "7.8",
        "description": "The MultipartStream class in Apache Commons Fileupload before 1.3.2, as used in
Apache Tomcat 7.x before 7.0.70, 8.x before 8.0.36, 8.5.x before 8.5.3, and 9.x before 9.0.0.M7 and other
products, allows remote attackers to cause a denial of service (CPU consumption) via a long boundary
string.",
        "publish_date": "2016-07-04T18:59:04.000Z",
        "references": [
          "http://svn.apache.org/viewvc?view=revision&revision=1743480",
          "http://svn.apache.org/viewvc?view=revision&revision=1743722",

        ...
        ],
        "summary": "CWE-20 Improper Input Validation"
      },
      "created": "2016-07-04T18:59:04Z",
      "affected_components": [
        {
          "id": "gav://org.apache.tomcat:tomcat-servlet-api:8.0.32"
        },
        {
          "id": "gav://org.apache.tomcat:tomcat-api:8.0.32"
        },

        ...
        }
        ],
  "top_artifacts": [
    {
      "name": "artifactory-pro-war-4.x-20160616.132515-1.war",
      "path": "/org/artifactory2/pro/artifactory-pro-war/4.x-SNAPSHOT/",
      "package_type": "Maven",
      "indexed": "2016-12-07T14:13:14Z",
      "vulnerabilities": [
        {
          "id": "584818fcaee4940008425415",
```

```
          "summary": "The authenticated-encryption feature in the symmetric-encryption implementation in the
OWASP Enterprise Security API (ESAPI) for Java 2.x before 2.1.0 does not properly resist tampering with
serialized ciphertext"
        },
        {
          "id": "5820903652a576000835b360",
          "summary": "CWE-399 Resource Management Errors"
        },

        ...
        }
        ],
  "lastUpdate": "2017-01-01T18:01:08.456580152Z"
}
```

Response Codes:

200: Security report successful

Get Recent Vulnerabilities

Description: Get recent vulnerabilities for a given week, for the security report
Security: Requires a valid user with "Generate Reports" permission
Usage: GET /securityReport/recentVulnerabilities?time=<End of week date in format: YYYY-MM-DD>&token=<authentication token>&direction=<asc l desc>&num_of_rows=<rows to display>&page_num=<page to view>
Produces: application/json

```
{
  "data": "interface",
  "total_count": 10
}
```

Sample Usage:

```
GET /securityReport/recentVulnerabilities?time=2016-12-
31&token=12345&&direction=asc&num_of_rows=20&page_num=1

{
  "data": null,
  "total_count": 0
}
```

Response Codes:

200: Successfully obtained all the vulnerabilities for a given week

400: Failed to get recent vulnerabilities - check pagination query parameters

400: Failed to get recent vulnerabilities - no time period was specified

---

Get Recent Components

Description: Get recent components for all security report in given week period
Security: Requires a valid user with "Generate Reports" permission
Usage: GET /securityReport/recentComponents?time=<End of week date in format: YYYY-MM-DD>&token=<authentication token>&direction=<asc l desc>&num_of_rows=<rows to display>&page_num=<page to view>

Produces: application/json

```
{
  "data": "interface",
  "total_count": 11
}
```

Sample usage:

```
GET /securityReport/recentComponents?time=2016-12-31&token=12345&&direction=asc&num_of_rows=20&page_num=1

{
  "data": null,
  "total_count": 0
}
```

Response Codes:

200: Successfully obtained recent components

400: Failed to get recent components - check pagination query parameters.

400: Failed to get recent components - no time period was specified

---

GRAFEAS

Grafeas is an open source API that enables comprehensive auditing and governance for your software supply chain. The API uses a unified metadata exchange format that creates a uniform and consistent way to produce and consume metadata from software components. Having a standardized API for working with metadata is great because it simplifies your workflow by promoting automation.

By fully supporting the Grafeas API, Xray allows you to combine the publicly available metadata and your private metadata together to get a complete picture.

Grafeas defines two key entities:

- **Notes:** An item or condition that can be found via an analysis or is used multiple times in a process. For example, a CVE could be the result of a vulnerability analysis of a Linux package. In a build process, we would store information about our builder in a note. In Xray, notes represent security vulnerabilities but in the future additional note types will be supported.
- **Occurrences:** Can be thought of as an instantiation of a note and describes how the note was found in a specific cloud resource or project (e.g., location, specific remediation steps, etc.), or what the results of a specific note were (e.g., the container images that resulted from a build). In Xray, occurrences represent concrete component issues.

Usage

The Xray Grafeas API uses the following syntax:

- **Project:** In Xray, the `project_id` has no significant meaning as the project scope does not exist. You can use any keyword, as we have used 'xray' in our examples.
- **Note:** Each note is represented by a unique note ID. This ID is required when querying notes as displayed in the following example: `/projects/<project_id>/notes/<note_id>`.
- **Occurrence:** There are two ways to query occurrences either by a specific note or by a component (filter query).
  The first option is to retrieve all occurrences for a specific note using the following syntax: `/projects/<project_id>/notes/<note_id>/occurrences`.
  Another option is to retrieve all occurrences for a specific component using the following syntax: `/projects/<project_id>/occurrences?filter=<url encoded component ID>`.

Authentication

The Grafeas REST API authentication uses an access token as a bearer token in an authorization header (`Authorization: Bearer`) with your access token.

Get Note

Description: Gets a note using the note ID.
Security: Requires a valid user with “View Components” permission.
Usage: GET /api/v1/projects/<project_id>/notes/<note_id>
Consumes: application/json
Sample Usage:

```
GET /api/v1/projects/xrayScanner/notes/XRAY-G3


{
                "shortDescription": "RHSA-2016:1776: java-1.6.0-openjdk security update (Important)",
                "longDescription": "The java-1.6.0-openjdk packages provide the OpenJDK 6 Java Runtime
Environment and the OpenJDK 6 Java Software...",
                "kind": "PACKAGE_VULNERABILITY",
                "attestationAuthority": {
                    "hint": {}
                },
                "vulnerabilityType": {
                    "severity": "Major",
                    "details": [
                        {
                            "package": "7:java-1.6.0-openjdk-devel",
                            "minAffectedVersion": {
                                "kind": "MINIMUM"
                            },
                            "maxAffectedVersion": {
                                "name": " 1:1.6.0.40-1.13.12.5.el7_2",
                                "kind": "MAXIMUM"
                            },
                            "severityName": "Major",
                            "fixedLocation": {
                                "version": {}
                            }
                        }
                    ]
                },
                "buildType": {
                    "signature": {}
                },
                "baseImage": {
                    "fingerprint": {}
                },
                "package": {},
                "deployable": {},
                "discovery": {},
                "relatedUrl": [
                    {
                        "url": "https://rhn.redhat.com/errata/RHSA-2016-1776.html"
                    }
                ],
                "createTime": "2016-11-27T09:14:08+02:00",
                "updateTime": "2016-11-27T09:14:08+02:00"
}
```

400: Notes associated with project_id do not exist.

Update Note

Description: Updates a note.
Security: Requires a valid user with “Manage Components” permission.
Usage: PUT /api/v1/projects/<project_id>/notes/<note_id>
Produces: application/json
Sample Usage:

```
PUT /api/v1/projects/xrayScanner/notes/XRAY-G3

{
     "name": "XRAY-G3",
     "shortDescription": "UPDATED - RHSA-2016:1776: java-1.6.0-openjdk security update (Important)",
     "longDescription": "The java-1.6.0-openjdk packages provide the OpenJDK 6 Java Runtime Environment and
the OpenJDK 6 Java Software Development
                         Kit.Security Fix(es):* An insufficient bytecode verification flaw was discovered in
the Hotspot component in OpenJDK. An untrusted
                         Java application or applet could use this flaw to completely bypass Java sandbox
restrictions. (CVE-2016-3606)* Multiple denial of
                         service flaws were found in the JAXP component in OpenJDK. A specially crafted XML
```

```
file could cause a Java application using JAXP
                        to consume an excessive amount of CPU and memory when parsed. (CVE-2016-3500, CVE-
2016-3508)* Multiple flaws were found in the
                        CORBA and Hotsport components in OpenJDK. An untrusted Java application or applet
could use these flaws to bypass certain Java
                        sandbox restrictions. (CVE-2016-3458, CVE-2016-3550) - update",
                "kind": "PACKAGE_VULNERABILITY",
                "attestationAuthority": {
                    "hint": {}
                },
                "vulnerabilityType": {
                    "severity": "Major",
                    "details": [
                        {
                            "package": "7:java-1.6.0-openjdk-devel",
                            "minAffectedVersion": {
                                "kind": "MINIMUM"
                            },
                            "maxAffectedVersion": {
                                "name": " 1:1.6.0.40-1.13.12.5.el7_2",
                                "kind": "MAXIMUM"
                            },
                            "severityName": "Major",
                            "fixedLocation": {
                                "version": {}
                            }
                        },
                        {
                            "package": "6:java-1.6.0-openjdk-src",
                            "minAffectedVersion": {
                                "kind": "MINIMUM"
                            },
                            "maxAffectedVersion": {
                                "name": " 1:1.6.0.40-1.13.12.6.el6_8",
                                "kind": "MAXIMUM"
                            },
                            "severityName": "Major",
                            "fixedLocation": {
                                "version": {}
                            }
                        },
                        {
                            "package": "7:java-1.6.0-openjdk-demo",
                            "minAffectedVersion": {
                                "kind": "MINIMUM"
                            },
                            "maxAffectedVersion": {
                                "name": " 1:1.6.0.40-1.13.12.5.el7_2",
                                "kind": "MAXIMUM"
                            },
                            "severityName": "Critical",
                            "fixedLocation": {
                                "version": {}
                            }
                        },
                        {
                            "package": "5:java-1.6.0-openjdk-src",
                            "minAffectedVersion": {
                                "kind": "MINIMUM"
                            },
                            "maxAffectedVersion": {
                                "name": " 1:1.6.0.40-1.13.12.4.el5_11",
                                "kind": "MAXIMUM"
                            },
                            "severityName": "Critical",
                            "fixedLocation": {
                                "version": {}
                            }
                        },
                        {
                            "package": "5:java-1.6.0-openjdk-demo",
```

```
            "minAffectedVersion": {
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.4.el5_11",
                "kind": "MAXIMUM"
            },
            "severityName": "Critical",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "6:java-1.6.0-openjdk-javadoc",
            "minAffectedVersion": {
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.6.el6_8",
                "kind": "MAXIMUM"
            },
            "severityName": "Critical",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "7:java-1.6.0-openjdk-javadoc",
            "minAffectedVersion": {
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.5.el7_2",
                "kind": "MAXIMUM"
            },
            "severityName": "Major",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "5:java-1.6.0-openjdk-devel",
            "minAffectedVersion": {
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.4.el5_11",
                "kind": "MAXIMUM"
            },
            "severityName": "Major",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "6:java-1.6.0-openjdk",
            "minAffectedVersion": {
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.6.el6_8",
                "kind": "MAXIMUM"
            },
            "severityName": "Major",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "6:java-1.6.0-openjdk-demo",
            "minAffectedVersion": {
```

```
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.6.el6_8",
                "kind": "MAXIMUM"
            },
            "severityName": "Major",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "7:java-1.6.0-openjdk",
            "minAffectedVersion": {
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.5.el7_2",
                "kind": "MAXIMUM"
            },
            "severityName": "Major",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "5:java-1.6.0-openjdk-javadoc",
            "minAffectedVersion": {
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.4.el5_11",
                "kind": "MAXIMUM"
            },
            "severityName": "Major",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "7:java-1.6.0-openjdk-src",
            "minAffectedVersion": {
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.5.el7_2",
                "kind": "MAXIMUM"
            },
            "severityName": "Major",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "6:java-1.6.0-openjdk-devel",
            "minAffectedVersion": {
                "kind": "MINIMUM"
            },
            "maxAffectedVersion": {
                "name": " 1:1.6.0.40-1.13.12.6.el6_8",
                "kind": "MAXIMUM"
            },
            "severityName": "Major",
            "fixedLocation": {
                "version": {}
            }
        },
        {
            "package": "5:java-1.6.0-openjdk",
            "minAffectedVersion": {
                "kind": "MINIMUM"
```

```
                            },
                            "maxAffectedVersion": {
                                "name": " 1:1.6.0.40-1.13.12.4.el5_11",
                                "kind": "MAXIMUM"
                            },
                            "severityName": "Major",
                            "fixedLocation": {
                                "version": {}
                            }
                        }
                    ]
                },
                "buildType": {
                    "signature": {}
                },
                "baseImage": {
                    "fingerprint": {}
                },
                "package": {},
                "deployable": {},
                "discovery": {},
                "relatedUrl": [
                    {
                        "url": "https://rhn.redhat.com/errata/RHSA-2016-1776.html"
                    },
                    {
                        "url": "https://access.redhat.com/security/cve/CVE-2016-3458"
                    },
                    {
                        "url": "https://access.redhat.com/security/cve/CVE-2016-3500"
                    },
                    {
                        "url": "https://access.redhat.com/security/cve/CVE-2016-3508"
                    },
                    {
                        "url": "https://access.redhat.com/security/cve/CVE-2016-3550"
                    },
                    {
                        "url": "https://access.redhat.com/security/cve/CVE-2016-3606"
                    }
                ],
                "createTime": "2016-11-27T09:14:08+02:00",
                "updateTime": "2016-11-27T09:14:08+02:00"
}
```


Response code:

400: Failed to update note.

Create Note

Description: Creates a new note.
Security: Requires a valid user with “Manage Components” permission.
Usage: POST /api/v1/projects/project_id>/notes/<note_id>
Consumes: application/json
Response: 201
Sample Usage:


```
 POST /api/v1/projects/xrayScanner/notes

{
                "shortDescription": "RHSA-2016:1776: java-1.6.0-openjdk security update (Important)",
                "longDescription": "The java-1.6.0-openjdk packages provide the OpenJDK 6 Java Runtime
Environment and the OpenJDK 6 Java Software...",
                "kind": "PACKAGE_VULNERABILITY",
                "attestationAuthority": {
                    "hint": {}
                },
```

```
                "vulnerabilityType": {
                    "severity": "Major",
                    "details": [
                        {
                            "package": "7:java-1.6.0-openjdk-devel",
                            "minAffectedVersion": {
                                "kind": "MINIMUM"
                            },
                            "maxAffectedVersion": {
                                "name": " 1:1.6.0.40-1.13.12.5.el7_2",
                                "kind": "MAXIMUM"
                            },
                            "severityName": "Major",
                            "fixedLocation": {
                                "version": {}
                            }
                        }
                    ]
                },
                "buildType": {
                    "signature": {}
                },
                "baseImage": {
                    "fingerprint": {}
                },
                "package": {},
                "deployable": {},
                "discovery": {},
                "relatedUrl": [
                    {
                        "url": "https://rhn.redhat.com/errata/RHSA-2016-1776.html"
                    }
                ],
                "createTime": "2016-11-27T09:14:08+02:00",
                "updateTime": "2016-11-27T09:14:08+02:00"
}
```


Delete Note

Description: Deletes a note.
Security: Requires a valid user with “Manage Components” permission.
Usage: DELETE /api/v1/projects/<project_id>/notes/<note_id>
Produces: application/json
Sample Usage:

```
DELETE /api/v1/projects/xrayScanner/notes/XRAY-G2
```

Response Code:

200: Note was deleted.
400: Failed to delete notes with project_id.

Get Occurrences by Note ID

Description: Gets all occurrences based on a Note ID.
Security: Requires a valid user with “View Components” permission.
Usage: GET /api/v1/projects/<project_id>/notes/<note_id>/occurrences
Consumes: application/json
Sample Usage:

```
GET /api/v1/v1alpha1/projects/xrayScanner/notes/XRAY-G3

{
                "shortDescription": "RHSA-2016:1776: java-1.6.0-openjdk security update (Important)",
                "longDescription": "The java-1.6.0-openjdk packages provide the OpenJDK 6 Java Runtime
Environment and the OpenJDK 6 Java Software...",
                "kind": "PACKAGE_VULNERABILITY",
```

```
                "attestationAuthority": {
                    "hint": {}
                },
                "vulnerabilityType": {
                    "severity": "Major",
                    "details": [
                        {
                            "package": "7:java-1.6.0-openjdk-devel",
                            "minAffectedVersion": {
                                "kind": "MINIMUM"
                            },
                            "maxAffectedVersion": {
                                "name": " 1:1.6.0.40-1.13.12.5.el7_2",
                                "kind": "MAXIMUM"
                            },
                            "severityName": "Major",
                            "fixedLocation": {
                                "version": {}
                            }
                        }
                    ]
                },
                "buildType": {
                    "signature": {}
                },
                "baseImage": {
                    "fingerprint": {}
                },
                "package": {},
                "deployable": {},
                "discovery": {},
                "relatedUrl": [
                    {
                        "url": "https://rhn.redhat.com/errata/RHSA-2016-1776.html"
                    }
                ],
                "createTime": "2016-11-27T09:14:08+02:00",
                "updateTime": "2016-11-27T09:14:08+02:00"
```


Get Occurrences by Component ID

Description: Gets all occurrences based on a URL encoded component ID.
Security: Requires a valid user with “View Components” permission.
Usage: GET /api/v1/projects/<project_id>/occurrences?filter=<url encoded component id>
Consumes: application/json
Sample Usage:

```
GET /api/v1/v1alpha1/projects/xrayScanner/occurrences?
filter=aHR0cDovL2tva286ODA4MC9jb21tb25zOmNvbW1vbnM6NC42OUAxMjM%3D


{
                "shortDescription": "RHSA-2016:1776: java-1.6.0-openjdk security update (Important)",
                "longDescription": "The java-1.6.0-openjdk packages provide the OpenJDK 6 Java Runtime
Environment and the OpenJDK 6 Java Software...",
                "kind": "PACKAGE_VULNERABILITY",
                "attestationAuthority": {
                    "hint": {}
                },
                "vulnerabilityType": {
                    "severity": "Major",
                    "details": [
                        {
                            "package": "7:java-1.6.0-openjdk-devel",
                            "minAffectedVersion": {
                                "kind": "MINIMUM"
```

```
                            },
                            "maxAffectedVersion": {
                                "name": " 1:1.6.0.40-1.13.12.5.el7_2",
                                "kind": "MAXIMUM"
                            },
                            "severityName": "Major",
                            "fixedLocation": {
                                "version": {}
                            }
                        }
                    ]
                },
                "buildType": {
                    "signature": {}
                },
                "baseImage": {
                    "fingerprint": {}
                },
                "package": {},
                "deployable": {},
                "discovery": {},
                "relatedUrl": [
                    {
                        "url": "https://rhn.redhat.com/errata/RHSA-2016-1776.html"
                    }
                ],
                "createTime": "2016-11-27T09:14:08+02:00",
                "updateTime": "2016-11-27T09:14:08+02:00"
}
```


Response Code:
400: Failed to get notes for project_ID.

SUMMARY

Build Summary

Description: Provides details about any build specified by build identifier (name + number)
Security: Requires a valid user with "View Components" permission
Usage: GET /summary/build?build_name=<build name>&build_number=<build number>
Produces: application/json


```
{
  "artifacts": [
    {
      "general": {
        "component_id": "",
        "name": "",
        "path": "",
        "pkg_type": "",
        "sha256": ""
      },
      "issues": [
        {
          "created": "",
          "description": "",
          "impact_path": [
            {}
          ],
          "issue_type": "",
          "provider": "",
          "severity": "",
          "summary": ""
        }
      ],
      "licenses": [
```

```
        {
          "components": [
            "sets.SetInterface"
          ],
          "full_name": "",
          "more_info_url": [
            ""
          ],
          "name": ""
        }
      ]
    }
  ],
  "errors": [
    {
      "error": "",
      "identifier": ""
    }
  ]
}
```

Response Codes:

200: Obtained artifact build summary

400: Missing build name or build number

Artifact Summary

Description: Provides details about any artifact specified by path identifiers or checksum
Security: Requires a valid user with "View Components" permission
Usage: POST /summary/artifact
Consumes: application/json

```
{
  "checksums": [
    ""
  ],
  "paths": [
    ""
  ]
}
```

Produces: application/json

```
{
  "artifacts": [
    {
      "general": {
        "component_id": "",
        "name": "",
        "path": "",
        "pkg_type": "",
        "sha256": ""
      },
      "issues": [
        {
          "created": "",
          "description": "",
          "impact_path": [
            {}
          ],
          "issue_type": "",
          "provider": "",
          "severity": "",
          "summary": ""
        }
      ],
```

```
      "licenses": [
        {
          "components": [
            "sets.SetInterface"
          ],
          "full_name": "",
          "more_info_url": [
            ""
          ],
          "name": ""
        }
      ]
    }
  ],
  "errors": [
    {
      "error": "",
      "identifier": ""
    }
  ]
}
```


Sample Usage:

```
POST /summary/artifact
{
    "checksums":["d160c68ed8879ae42756e159daec1dd7ecfd53b6192321656b72715e20d46dd2"]
}

Response
{
  artifacts: [
    "general": {
      "name": "artifactory-pro.zip",
      "path": "art2/ext-release-local/",
      "pkg_type": "Generic",
      "sha256": "d160c68ed8879ae42756e159daec1dd7ecfd53b6192321656b72715e20d46dd2",
      "component_id": "gav://org.artifactory.pro:artifactory-pro-war:4.14.0"
    },
    "issues":[
      {
        "summary":"FileSystemBytecodeCache in Jinja2 2.7.2 does not properly create temporary directories",
        "description":"this is the description of the issue",
        "issue_type":"security",
        "severity":"Major",
        "provider":"JFrog",
        "created":"2016-10-26T11:15:51.17Z",
        "impact_path": [
          "xray-artifactory/maven-1000/com/atlassian/aui/auiplugin/0.0.5-9-0-snapshot-035-do-not-use/Jinja2-
2.7.2"
        ]
      }
    ],
    "licenses":[
      {
        "name":"MIT",
        "full_name":"The MIT License",
        "more_info_url":"https://opensource.org/licenses/MIT",
        "components":[
          "some-component-1",
          "some-component-2",
          "some-component-3"
        ]
      },
      {
        "name":"AGPL-3.0",
        "full_name":"GNU AFFERO GENERAL PUBLIC LICENSE, Version 3",
        "more_info_url":"https://opensource.org/licenses/AGPL-3.0",
        "components":[
```

```
          "some-component-4",
          "some-component-5"
        ]
      },
      {
        "name":"unknown",
        "components":[
          "some-component-6",
          "some-component-7"
        ]
      }
  ],
  errors: [
    {
      identifier: "4e39f19212597312ee02db873847bcb12c17cc639898bd2fd9b6a4aff16690e5",
      error: "Artifact doesn't exist or not indexed in Xray"
    }
  ]
}
```

Response Codes:

200: Obtained artifact summary

415: Failed to parse JSON

SYSTEM

Ping Request

Description: Sends a ping request
Security: Requires a valid user
Usage: GET /system/ping
Produces: application/json
Sample usage:

```
GET /system/ping
{"status":"pong"}
```

Response Codes:

200: Ping successful

External Ping Request

Description: Sends a ping request to external sources (Global Database, Bintray, etc.)
Security: Requires a valid user
Usage: GET /system/external/ping

Response Codes:
200: Ping successful to external source
404: Page not found

Get Version

Description: Gets the Xray version and revision you are running
Security: Requires a valid user
Usage: GET /system/version
Produces: application/json

```
{
    "xray_version":"<version number>",
    "xray_revision":"<revision number>"
}
```

Sample Usage:

```
GET /system/version

{
    "xray_version":"1.4",
    "xray_revision":"b3034"
}
```

Response Codes:

200: Got version info successfully

Troubleshooting

Overview

This page provides tips to solve common problems that users have encountered.

Configuration Folder

In several cases, the solution requires accessing Xray internal data and configuration files. The location of these files can be obtained by running `./xray info`.

For example:

```
./xray info

server port: 8000
docker mount folder: /home/vagrant/.jfrog/xray
```

Page contents

Xray is not indexing artifacts

Cause	There may be a problem with the RabbitMQ message broker
Resolution	Check the RabbitMQ event, index, or persist queues for messages. You can access the queues through the RabbitMQ console using: http://localhost:15672/#/queues If you are not able to acess the RabbitMQ UI, you can try to create an SSH tunnel using: ssh -L15672:127.0.0.1:15672 root@<machine ip>

Cause	Xray has reached the configured limit for disk usage (the default is 80%)
Resolution	Increase the disk usage limit by increasing the value of `maxDiskDataUsage` in Xray's *`xray_config.yaml`* configuration file.

Xray is not able to communicate with Artifactory

Cause	Xray is not configured to work in insecure mode
Resolution	Enable insecure mode by setting the following parameter in Xray's *`xray_config.yaml`* configuration file. `sslInsecure: true`

Xray does not not load the web UI or interact with the REST API

Cause	Typically, this could be caused due to the fact that one or more of Xray's microservices are not started

Resolution	Use the Xray sh script to check the status of the microservices and verify all of them are fully started and up, e.g.:
	
	`./xray.sh status all`
	
	Or for Docker:
	
	`./xray.sh ps`

Xray's RabbitMQ's queue has a large number of messages

Cause	A working Xray system should have its RabbitMQ's queues' count be relatively levelled (close to 0) which would signify and provide the best performance and fastest results. Large could of messages may happen be due to either: 1. One Xray service/component is not working fast enough to handle its queue 2. Some error/unexpected behavior happened on one of the components, searching for the logs according to the corresponding RabbitMQ queue will provide an insight
Resolution	1. Temporarily increase the number of workers responsible for the appropriate queue (e.g. analysis) or allocate more resources to the machine; RAM & CPUs for faster processing, SSD for faster disk I/O. 2. Search the appropriate log file (e.g. xray-analysis.log) for any [EROR] or "Caused by" messages for the cause.

Xray is utilizing a large amount of space and potentially halts indexing

Cause	This may happen be due to either (or a combination of): 1. The $XRAY_DATA location is not configured correctly - thus preventing the Xray disk sampler from checking the correct path for disk space availability 2. The allowed disk utilization percentage value is too high 3. Temporary files which should be removed (under $XRAY_HOME/data/data/), are consuming a large amount space 4. * **Mostly** applicable for large scale: Other databases (RabbitMQ/PostgreSQL/MongoDB) are occupying a large amount of space

Resolution	1. Enable debug logging on the Event component to see what is the path checked, expect to see a log entry stating: "Data Folder <location> , used: <%>, configured Threshold <%>. Remedy this by altering the xrayDataFolder property under the *xray_config.yaml* file to a correct location.
	
	2. Configure under the *xray_config.yaml* the maxDiskDataUsage flag to a lower value. On large scale enviroment and especially when working with large files, more space will be consumed (temporary) to allow the recursive indexing process to complete.
	
	3. Xray version 1.9 significantly improved on handling scenarios where indexing has failed due an unexpected error (such as one related to files corruption) and will automatically remove these.
	
	It is also possible to check the RabbitMQ's **retry** queues content (Get Message) to inspect messages and correlate their location under the xray_indexer.log* files. This will provide an insight for the retry action.
	
	4. Configure the RabbitMQ database (mnesia) along with $XRAY_HOME (**the two should reside under the same mount)** to a large mount (typically 1-2TB)
	
	/etc/rabbitmq/rabbitmq-env.conf (not created by default):
	
	RABBITMQ_MNESIA_BASE=/home/rabbitmq/mnesia
	
	Separate the Mongo and PostgreSQL (can and/or should be paired under the same mount but separate) from the RabbitMQ & $XRAY_HOME:
	
	Postgres:
	
	• To check: Postgresql home data folder location is configurable using the installer
	
	If to be moved after the installation completion it is possible to follow this guide - **Make sure PostgreSQL is stopped**
	
	Mongo:
	
	• **Stop Mongo** (preferably stop all Xray services using the xray.sh stop all command)
	• edit **/etc/mongodb.conf,** change**:**
	
	`dbpath=/var/lib/mongo`
	
	to a path of your choice.
	
	• Copy your mongo data folder to a new location - make sure the permissions are set with the mongod user (the -a flag of the cp command should take care of keeping permissions)
	
	`cp –ra /var/lib/mongo  /home/myuser/data/mongo`

Xray does not startup after a system failure incident or disk space exhaustion

Cause	If RabbitMQ is not started successfully (use the commands from the troubleshooting above), it could be that RabbitMQ ran into data corruption after an unexpected system state occurred and caused its messages' and/or queues' storage to loss integrity.

Resolution	• Please note that this could result in queued messages loss - therefore some already made processing by Xray's microservices will be lost. **No data related to already indexed files persisted to PostgreSQL (where the impact graph is stored and built) will be affected.**
	
	Check the following RabbitMQ files to diagnose the reason:
	
	/var/log/rabbitmq/startup_log
	
	/var/log/rabbitmq/startup_err
	
	These log files are useful in particular when trying to diagnose RabbitMQ related issues.
	
	• Typically commands such as *$ head* or *$ tail* would help considerably in troubleshooting of these.
	
	Below is a common error indicating a *bad_match* error (excerpted):
	
	`BOOT FAILED`
	`==========`
	
	`Error description:`
	`{could_not_start,rabbit,`
	`{{badmatch,`
	`{error,`
	`{{{badmatch,`
	`{error,`
	`{not_a_dets_file,`
	`"/var/lib/rabbitmq/mnesia/rabbit@<id>/recovery.dets"}}},...`
	
	In this case the removal of RabbitMQ's queues' related files under the path below would help:
	
	/var/lib/rabbitmq/mnesia/<username>@<id>/recovery.dets
	
	Then try to restart Xray's microservices using:
	
	$ ./xray.sh restart all
	
	• Sometimes, the restart of RabbitMQ alone could suffice (e.g. using $ service rabbitmq-server restart)
	
	If this does not help, carry on to remove the queues message stores directors:
	
	/var/lib/rabbitmq/mnesia/<username>@<id>/msg_store_transient
	
	/var/lib/rabbitmq/mnesia/<username>@<id>/msg_store_persistent
	
	/var/lib/rabbitmq/mnesia/<username>@<id>/queues
	
	Again, restart Xray's microservices and access Xray using the web UI.

Xray is not able to send email notifications

Cause	Xray is not configured to work with an email server in insecure mode
Resolution	Enable insecure mode by setting the following parameter in Xray's *`xray_config.yaml`* configuration file . . `sslInsecure: true`

Xray is not providing enough information in its log files

C	By default, Xray is configured for INFO level logging.

Cause	
Resolution	For each log file you want to analyze, set the log level to DEBUG. The log level for each Xray service is set in a corresponding configuration file found under `/var/opt/jfrog/xray/data/config` (for a Linux installation). For example, the analyzer service debug level is set in `analysis_log4go_config.xml`, while the indexer service debug level is set in `indexer_log4go_config.xml`. Similarly for all the other Xray services.

Xray database migration failed and needs to be retried

Cause	After performing an upgrade, Xray is performing a migration. When the migrations starts, it creates a lock mechanism, so that external operations on the database will be prevented, while performing the migration. In case a migration has failed, the lock will remain, locking Xray, but it can be retried and can be successful on next run.

Resolution	**Quick summary of the steps we are about to take:**
	
	1. Check why and which the migration failed. This is optional and although not mandatory but is useful for future avoidance.
	
	2. Delete the lock so that the migration will be retried.
	
	Here is a break down of the steps that will need to be taken:
	
	1. Stop all Xray services:
	
	Non-Docker
	
	$ ./xray.sh stop all
	
	Docker:
	
	$ ./xray.sh stop
	
	2. Start MongoDB and PostgreSQL only:
	
	Non-Docker:
	
	$ service postgresql-9.5 start
	
	$ service mongod start
	
	Docker:
	
	$ docker start xray_mongodb_1
	
	$ docker start xray_postgres_1
	
	3. Perform actions on **MongoDB** side:
	
	a. Connect to MongoDB:
	
	Non-Docker:
	
	$ mongo -u xray -p password --authenticationDatabase xray
	
	Docker:
	
	$ docker exec -it $xray_mongodb_1 bash
	
	Inside the Docker bash shell connect to MongoDB:
	
	$ mongo -u xray -p password --authenticationDatabase xray
	
	b. Switch to Xray DB:
	
	$ use xray

c Find the current state of DB migrations, including the failed ones (important for understanding the DB schemas migration versions):

```
$ db.db_migrations_running.find({}).pretty()
```

e. Copy the output aside for investigation:

```
$ db.db_migrations.find({}).sort({"version":-1}).limit(1)
```

example for the output (**do this for the 4. analysis step below**):

{ "_id" : ObjectId("5a4bd9be95e701000100c37b"), "version" : NumberLong(27) }

* The line above will tell us the ID of the migration running

f. Remove all migrations locks with the following command:

```
$ db.db_migrations_running.deleteMany({})
```

At this point - you may skip to step 5.

4. For analysis and when contacting JFrog Support, the following information needs to be collected:

Database state only information:

a. MongoDB, copy the the output of two commands (were provided in step 3 above):

a. $ db.db_migrations_running.find({}).pretty()

b. $ db.db_migrations.find({}).sort({"version":-1}).limit(1)

b. PostgreSQL, collect the information below from PostgreSQL database:

Connect to the PostgreSQL database:

Non-Docker:

```
$ psql xraydb xray
```

Docker:

```
$ docker exec -it  xray_postgres_1 /bin/bash
$ psql xraydb xray
```

* If/When prompted for password, enter *xray*

* *xraydb* is the name of the db, *xray* is the username

a. Run the following query (note to use ';' in the end, without it, it will not execute):

```
$ SELECT * FROM schema_migrations;
```

* Copy the output aside

b. In order to exit, enter:

```
$ \q
```

c. Xray Services logs from the migration timeframe:

xray_server.log

	xray_persist.log xray_event.log 5. **Restart all Xray services. This will trigger the migration to re-run.**

Release Notes

Overview

This page presents release notes for JFrog Xray describing the main fixes and enhancements made to each version as it is released.

Download

Click to download the latest **Xray.**

Installation

For installation instructions please refer to Installing Xray.

Xray 1.12

Released March 28, 2018

Highlights

Improved Integration with Artifactory

JFrog Xray 1.12 jointly released with JFrog Artifactory 5.10, presents significant changes in how these two complementary applications are integrated to improve usability and stability.

> **Upgrade Xray first**
>
> For this joint release of JFrog Artifactory 5.10 and JFrog Xray 1.12, we strongly recommend first upgrading your Xray installation to version 1.12 and only then upgrading Artifactory.

Scan Status

Previously, an artifact's scan status was stored in Artifactory by annotating the artifact with a set of properties such as indexing status, last update, top vulnerability severity and block status. From this version, these properties will be removed. Artifactory will fetch an artifact's scan status on demand when it is selected in the tree browser.

This is a breaking change which restricts compatibility of Artifactory and Xray versions as described in the following table:

		Xray Version	
		1.12+	**<1.12**
Artifactory Version	**5.10 +**	Since both Artifactory and Xray are upgraded, the new integration is fully functional as designed.	This combination is supported. Artifactory will continue to display each artifact's scan status, however, it will use previous mechanism that uses properties.
	<5.10	In this combination, the integration will not work since an older version of Artifactory does not query Xray for scan status, and the new version of Xray does not attach properties to an artifact.	If neither Artifactory nor Xray are upgraded, the integration will work using the previous mechanism that displayed scan status as a set of properties on the artifact.

Improved Download Blocking

From this release, download blocking has been removed from Artifactory and is, instead, configured in Xray as "Block Download" action on a Watch. This creates a more intuitive and consistent workflow giving you full control over all actions on an Artifact that has a violation in one place.

Better Build Control

Page Contents

Previously, all builds in Artifactory were indexed by Xray potentially causing visual clutter as less important builds such as snapshots would get indexed. From this version, Xray lets you select which builds to index, letting you focus your analyses on more significant build processes.

> **Reapply Indexing to Your Builds**
>
> During the upgrade process to Xray 1.12, the indexing is cleared from all of the builds. To reapply indexing, you need to explicitly apply indexing to the builds of your choice.

Component-Driven Workflow: Watch Violations

Previously, Xray brought vulnerabilities to your attention in the form of Alerts. But since alerts may aggregate several issues, each of which may affect multiple artifacts and builds, they made it difficult to understand all the issues affecting a particular component. Continuing Xray's evolution to a component-driven workflow, this version introduces **Watch Violations**.

Watch violations are displayed directly on the Component Details panel making it easy to identify all the security, license and custom issues affecting the component.

Support for Additional Package Types

From this version, Xray supports indexing and scanning Gradle, Ivy and SBT packages.

Xray 1.12.1

Released April 2, 2018

Issues Resolved

This patch fixes these issues that were discovered in version 1.12:

1. Fixed an issue in which the “Block Download” action did not work when creating a watch on a remote repository when Artifactory versions lower than 5.10.
2. Fixed an issue in which deleting a custom issue did not impact the “Block Download” action when using Artifactory versions lower than 5.10.
3. Added the ability to automatically index all builds by Xray via a configuration parameter.

Xray 1.11

Released February 18, 2018

Highlights

Assign OSS Licences to Components

This release provides these advanced **OSS licenses functionalities**:

- Assign custom licenses to your components.
- View the source of the originated license - custom, JFrog, or a local file.
- View if the license was assigned directly to the component or propagated to parent components and is part of their license list.

Failure Messages

Xray now displays all impact analysis and artifact scanning failures in the new **Failure Messages page**, in the Admin module. This page provides administrators a single place where they can easily identify the exact step in the scanning and impact analysis Xray process in which it failed, allowing them to fix the issue and retry the step.

Xray 1.10

Released January 2, 2018

Highlights

Grafeas API

Project Grafeas defines an open, unified metadata exchange format and API that will create a uniform and consistent way to produce and consume metadata from software components. By fully supporting the Grafeas API, Xray acts as a portal to Grafeas providing your software supply chain with an unprecedented abundance of metadata that can be easily be put to use in automated auditing and governance processes. This release of Xray exposes a set of Grafeas endpoints that are fully integrated into the Xray REST API.

Externalization of Databases

Xray works with a number of third party services, such as various databases, which were previously pre-installed with the other Xray microservices. From Xray 1.10, you have more control over your resource allocation and you can direct Xray to use an external RabbitMQ, MongoDB or PostgreSQL database in use in your organization. Keep in mind that if you direct Xray to use an external database, you have full control over the database, and also full responsibility to maintain and backup the database for Xray's use.

Improved Database Sync

Database synchronization has been significantly improved resulting in a smoother workflow, data compression and boosted performance. The enhanced compression and performance promote stability and robustness to transient network issues. Depending on your hardware, network and other factors, this may improve performance by up to 70%.

Expanded IDE Integration

In addition to UI improvements, the Unknown macro: 'link' has been updated to support scanning of Gradle and npm package formats in addition to the existing Maven package format.

Issues Resolved

1. Improved performance of Alerts page.
2. Fixed an issue whereby Ignore Rules were not fetched when their associated Watch was deleted.
3. Fixed an issue whereby the Event microservice crashed due to a missing checksum returned from a property search made for a build's items.
4. Fixed an issue whereby deployed Docker Images with same SHA256 value got stuck in the analysis retry queue.
5. Fixed an issue whereby the Xray API 'Update User' command does not update the user, and returns a 404 error.

Xray 1.10.1

Released January 4, 2018

This release includes UI display issue fixes in the Integrations and Permissions pages.

Xray 1.9

Released December 3, 2017

Highlights

Authentication Through External Providers (LDAP, SAML, Crowd etc.)

User management in Xray has been greatly simplified by adding the ability to authenticate users through your corporate LDAP/Crowd or SAML provider. All you need to do is define one of the connected Artifactory instances as an "Authentication Provider". This lets you import the LDAP/Crowd, SAML and internal Artifactory users and groups from the specified Artifactory instance to Xray, and then assign them permissions as needed.

Permission Management

This version introduces a flexible permissions model that gives an administrator fine-grained control over how users and groups access the different features of Xray. "Resources" define the scope of a permission and specify the repositories and builds in the connected Artifactory instance to which the permission applies. You can then specify users and groups, internally defined in Xray or imported from a connected "authentication provider" as described above, and grant them privileges for the selected resources.

Feature Enhancements

1. Improved performance of indexing and analysis for large-scale environments.
2. Improved indexing and analysis process for Javascript files.
3. Improved database processes which significantly improve performance of certain recursive queries

Issues Resolved

1. Fixed an issue in which Xray would not delete files that were moved to the RabbitMQ failure queue.
2. Fixed several issues connected to identifying the OS layer and its installed packages in Docker images.
3. Fixed an issue that would cause RabbitMQ to drain available RAM on large scale environments by loading a large number of messages.

Xray 1.8

Released July 13, 2017

Highlights

Content-Driven Workflow

Xray's current work flow is event-driven creating alerts with stateless information; a snapshot of builds and components at an instant in time. In this release, we are adding support for a new and more intuitive workflow which is content-driven in that issues are displayed based on the **components** y ou are interested in. This has a huge impact on how you navigate your way to the most relevant content. The high-level flow can be summarized as:

Search for components Drill down Examine issues

Enhanced Search: Xray now provides enhanced search allowing you to search for specific components through a set of search filters such as package type, issue severity, version and more.

Rich component display: From the search results, you can select the component that interests you and view a rich display that provides details of all versions of the selected component

Examine issues: Selecting any issue from the components display provides detailed information on the issue as well as a list of all the artifacts and builds on which it has an impact.

Recommendations for Remediation

In addition to providing a comprehensive list of versions in which a vulnerability exists for an infected component, the rich component display in the content-driven workflow also indicates in which version a vulnerability has been fixed (if available) and recommends upgrading to that version

JFrog IntelliJ IDEA Plugin

With the JFrog IntelliJ IDEA Plugin, you can scan your Maven project dependencies using Xray and view vulnerabilities **during development time** dire ctly from within the IntelliJ IDE. IDE integration support will continue to expand to additional industry-standard IDEs, and to additional package formats.

TeamCity Integration

JFrog Xray expands its CI/CD integration capabilities by adding support for TeamCity, enabling you to scan builds, generate reports and even fail build jobs if they use components with known vulnerabilities. This is an effective way to prevent builds with vulnerabilities from entering production systems.

Enhanced Vulnerability Data

Processing of raw vulnerability data has been greatly enhanced based on improved algorithms and heuristics to correlate and match data from different sources to the right component and version. This new data model provides greater and more accurate details about vulnerabilities such as infected version ranges, fix versions and more. It also allows better identification of infected components. In the case of Maven components, the vulnerability data has been completely replaced and undergoes manual curation before being loaded into the database resulting in better coverage with fewer false-positives.

Note that you need to perform a database sync (whether you are working in online or offline mode) to work with the enhanced vulnerability data.

Feature Enhancements

Improved Scanning Performance

Performance of scanning new builds and artifacts has been dramatically improved to orders of magnitude. Since this is the most common process that Xray performs the improvement results in Xray being more responsive on the whole. In particular, the performance and accuracy of Docker images analysis has been greatly improved.

Support for Docker Images in Builds

Docker images encased in builds are now scanned and indexed just like any other build dependency.

Issues Resolved

1. Fixed an issue in which Xray listed a component as having an "Unknown" license, even though specific known licenses were identified.
2. Fixed an issue in which npm dependencies with vulnerabilities were downloaded even when their hosting repository in Artifactory was set to block downloads.

Xray 1.8.0.1

Released July 17, 2017

Issues Resolved

1. Fixed an issue which may have caused a slow database migration when upgrading Xray to a new version.

Xray 1.8.1

Released July 31, 2017

Issues Resolved

1. Fixed an issue with Xray's analysis process causing component license data to be saved multiple times, potentially consuming high amounts of memory and disk space.
2. Fixed an issue where license issues in alerts did not have an impact path.
3. SaaS users will now receive email notifications with a default mail server configuration.

Xray 1.8.2

Released August 3, 2017

Issues Resolved

1. Issues and their status, contained within Docker images in a build, are now properly propagated.

Xray 1.8.3

Released August 22, 2017

Feature Enhancements

1. Xray now gives you the option of selecting a custom location for your **Xray data** and **PostgreSQL** directories during an installation or upgrade process.
2. Xray has undergone system-wide performance improvements which can be seen in several screens including Components, Issues in component details, Alerts, Security Reports and more.
3. When viewing component details, you can filter the issues displayed by their **Summary** field.
4. The Reports module has undergone several improvements in UI display and performance

Issue Resolved

1. Fixed an issue in which the **All Alerts** tab in the Alerts screen would appear empty even when alerts were present.

Xray 1.8.3.1

Released August 24, 2017

Issues Resolved

1. Fixed an issue where some filter selections in the Component Search did not return all applicable results.
2. Fixed an issue with the “Cancel” button in the Artifactory instance details page.
3. Fixed an issue where data migration was not running properly when upgrading to Xray 1.8.3 resulting in many errors in the log file.

Xray 1.8.4

Released September 25, 2017

This release brings significant OSS licenses functionalities for improved license coverage, including the ability to parse license from files, license content analysis and GitHub license matching.

Feature Enhancements

1. Xray will now support parsing OSS license information from all popular license file conventions, such as "*.pom" and "license.txt" metadata files.
2. An additional layer of matching logic will now be used to help classify even more OSS licenses that may have been slightly modified, by analyzing the license content and comparing it to known license types.
3. Xray is now able to get license information from GitHub for components with a GitHub page.
4. The Xray installer now supports writing the installation / upgrades outputs to an installer log for better traceability.

Issues Resolved

1. The license tab will now show a "0" in the tab header when a license cannot be identified. Licenses that are identified as "unknown" will include a proper placeholder in the component details page.
2. Better handling of multiple files associated with the same component id.
3. The Xray Docker installation does no longer require root privileges to run.

Xray 1.8.5

Released October 17, 2017

Feature Enhancements

1. **Artifact Checksum Matching** - Xray now provides more accurate results by doing a checksum match in addition to the already supported component id match. This is especially useful for files which do not have proper component id's attached to them, such as Javascript files.
2. **Performance improvements in the analysis process and Memory Consumption Enhancements**.

Xray 1.8.6

Released October 19, 2017

Issues Resolved

1. Fixed an issue in which when Xray's indexing process would fail, under certain conditions, temporary files would not be removed which could eventually deplete available storage on the filesystem.

Xray 1.7

Released April 20, 2017

Highlights

New Home Page: The Xray Home page has been completely redesigned to act as a dashboard that provides a wealth of useful information. At a glance, understand your general system health, get an overview of components and alerts, system scan status, database sync status and more.

Feature Enhancements

Package Type filter for Component search: The Components page now includes a Package Type filter that lets you focus on specific package types making it easier for you to search for specific components.

Issues Resolved

1. If an external integration is removed, Xray will now also remove any alerts related to that integration
2. Custom issues are now aggregated together with security vulnerabilities when viewing Component details and in REST API responses.
3. Fixed an issue with updating properties in Artifactory that are related to Xray's indexing status.

Xray 1.7.1

Released April 24, 2017

Issues Resolved

1. Fixed an issue with file paths that sometimes led to the wrong location.
2. Fixed an issue with migration for component license migration.

Xray 1.7.2

Released June 5, 2017

Feature Enhancements

1. Xray now adds a timestamp indication to build snapshots. This ensures that each snapshot will have a unique name, making it easier to work with snapshots.
2. When updating to a new version requires migration of the database (which may take some time), Xray will now show how the upgrade is progressing and provide error information if the upgrade fails.
3. Xray's logging facility has been improved so that you no longer have to restart Xray if you want to change the log level for any of it's services.
4. Xray's search has been enhanced so that in addition to package type, you can now also filter searches by component type (artifacts or builds).

Issues Resolved

1. Fixed an issue that prevented creation of custom issues due to an error in parsing the timestamp when it included the Z timezone indicator.
2. Fixed an issue that prevented Xray from annotating artifacts in Artifactory whose name included certain special characters.
3. Fixed an issue in which the Xray base URL in the config descriptor for a connected Artifactory instance would not be updated when the base URL was modified.
4. Fixed an issue in which the status of some artifacts would not be modified even after they were scanned, and, as a result, their download was blocked when download blocking was enabled in Artifactory for unscanned artifacts.

Xray 1.7.2.1

Released June 6, 2017

Issues Resolved

1. Fixed an issue introduced in version 1.7.2 which, under certain conditions, caused a database connection leak.

Xray 1.7.2.2

Released June 8, 2017

Issues Resolved

1. Fixed an issue that prevented Xray from synchronizing its database and indexing artifacts due to too many idle connections to its PostgreSQL database.

Xray 1.7.3

Released June 25, 2017

Enhancements

Xray now supports setting the system log level for each of the microservices without having to restart the Xray server.

Issues Resolved

1. Fixed an issue in which Docker images, whose full set of layers were already included in another indexed image, would not get indexed.
2. Fixed an issue in which the Artifact Summary REST API endpoint did not provide license information if there were no Allowed or Banner License filters defined for a watch.

Xray 1.6

Released January 18, 2017

CI/CD Integration

JFrog Xray takes an active role in your CI/CD pipeline to indicate you should fail build jobs if your build or any of its dependencies have vulnerabilities. Your CI server (currently, Jenkins CI is supported) can now send a request to Xray to scan a build that was uploaded to Artifactory. In accordance with Watches you may define, Xray will scan the build, and if vulnerabilities that trigger an alert are found, Xray can now respond to the inquiring CI server that the build job should fail.

Main Updates

1. Fail build jobs according to Watch specifications if build artifacts or their dependencies contain vulnerabilities.
2. Changes in the UI for Watches replacing "Notifications" with "Actions", and the addition of the Fail Build Job action to support CI/CD integration
3. "All Builds" has been added a new target type for watches so you can specify that all builds uploaded to Artifactory are scanned by Xray, not only specific builds you configure into the Watch.

Xray 1.6.1

Released January 25, 2017

Main Updates

1. An issue that was causing artifact indexing to fail has been fixed.

Xray 1.6.2

Released February 12, 2017

Preventing Brute Force Attacks

Xray has been equipped with a login protocol to prevent brute force attacks. When Xray encounters multiple login attempts by the same user, Xray steadily increases the time interval that the user must wait before attempting login again. After a specific number of failed login attempts, the user will be locked out of their account. At that point, login can only be reset by an Xray administrator. The administrator has full control over the number of failed login attempts to lock the user out.

System Logs

An Xray administrator may now view the Xray system log file in the **Admin** module, with the ability to filter log messages from the different services behind Xray.

Main Updates

1. A bug preventing Xray from reaching the global database server when a proxy server is configured was fixed.
2. Performance when synchronizing the global database to Xray has been greatly improved. The overall process time is dramatically reduced, both for a first-time synchronization, and for periodic updates.
3. A mechanism has been added to prevent brute force attacks on Xray by locking out users with multiple failed login attempts.
4. A bug that prevented upgrade when the upgrade archive was extracted in the same folder as the previous version, has been fixed.
5. The impact path of an artifact is now displayed as a full path including the Artifactory instance and the repository in which the impacted artifact is hosted.
6. The Xray log can now be viewed by an administrator in the **Admin** module System Logs page.

Xray 1.6.3

Released March 7, 2017

Main Updates

1. Xray's analysis process performance has been greatly improved
2. Performance when generating a security report has been greatly improved, especially for Xray instances that have indexed thousands of artifacts.
3. Alerts can now be sorted by severity, and when viewing the details for a selected alert, the tab title also displays its severity.
4. A bug in which some impacted artifacts were omitted from the security report has been fixed.
5. A bug in which offline database sync was failing due to components not being found has been fixed.
6. The scanning process performance has been greatly improved
7. When viewing a component's details page, vulnerabilities and licenses of it's child components are also displayed.

Xray 1.6.4

Released March 14, 2017

Main Updates

1. An issue causing proxy server functionality to fail has been fixed.

Xray 1.6.5

Released March 22, 2017

Main Updates

1. Improve performance of both the indexing and scanning processes.
2. Improve performance of security report generation.

Xray 1.5

Released January 4, 2017

Dependency Graph APIs

JFrog Xray exposes its dependency graphs to any external source with access to its REST APIs. Through a simple REST API call, you can now receive the full dependency graph of any component or build as a JSON object, or compare the dependency graphs of any two components or builds to get a clear indication of the differences between them and easily hone in on new dependencies that may have introduced issues and vulnerabilities.

Editing System Watches

System watches are created when a repository in Artifactory has been configured to block downloads. To provide more flexibility and finer control over when alerts should be generated, system watches can now be edited by Xray admin users.

Unknown Licenses

Handling components with unknown licenses is a matter of your organization's policy. Xray now allows you to specify if these components should trigger alerts or not.

Main Updates

1. Dependency Graph APIs allowing you to get the graph of any artifact or build, and compare any two artifacts or builds.
2. System watches can now be edited by Admin users.
3. Allowed and Banned License filters now allow you to specify "Unknown" so you can decide if components with unknown licenses should trigger alerts or not.
4. When indexing Docker images, Xray now also indexes Debian and RPM packages in the image OS layer.
5. The onboarding wizard UI has been improved for usability and to allow indexing selected repositories on the spot.
6. The Security Report display has been improved.

Xray 1.5.1

Released January 9, 2017

Main Updates

1. Fixed an issue that caused a database connection leak
2. Fixed handling of gzip files with invalid headers

Xray 1.5.2

Released January 10, 2017

Main Updates

1. Fixed an issue that prevented microservices from writing entries to system logs

---

Xray 1.4

Released December 20, 2016

Security Reports

JFrog Xray adds a new report that shows you which vulnerabilities have the most far reaching consequences in your code, and which components in your code base have the most reported vulnerabilities, as well as recent vulnerabilities and infected components that were detected.

Black Duck Integration

JFrog Xray has integrated with Black Duck Software as a new external vulnerability provider. Black Duck automates the process of securing and managing open source software by helping you comply with open source license requirements and providing security alerts about vulnerabilities discovered in open source components.

Main Updates

1. Security Report
2. Black Duck integration

Xray 1.3

Released December 4, 2016

Improved Onboarding

The onboarding experience has been improved in several ways including a wizard that guides you through the first essential steps of configuring Xray.

Integrations

The Integrations UI has been modified to be more flexible and efficiently accommodate any number of integrations with external issue and vulnerability providers.

Artifact and Build Summary REST API

The Artifact and Build Summary REST API endpoints provides general information about an artifact or build as well as an aggregated list of issues and OSS licenses associated with them.

Main Updates

1. Onboarding improvements including an Onboarding wizard
2. Flexible UI for integrations
3. Artifact Summary and Build Summary REST APIs

Xray 1.2

Released November 6, 2016

License Reports

Generate a report that shows the distribution of open source licenses used by artifacts indexed by Xray, as well their compliance with "Allowed Licenses" and "Banned Licenses" filters defined in all watches in the system.

System Status

Xray now monitors a variety of system parameters and reports on their status to let you easily diagnose problems.

Issue Filters

You can now create filters on watches based on the minimum severity of issues associated with indexed artifacts.

Main Updates

1. License reports for distribution of OSS licenses and compliance with watches defined.
2. Self-monitoring system status
3. Checksum calculation has been optimized by running it asynchronously.
4. Issue filters based on minimum severity of an issue associated with an artifact.

Xray 1.1

Released September 22, 2016

Support for Older Versions of Artifactory

JFrog Xray now supports all versions of JFrog Artifactory from v4.0 and above

Synchronization with the Global Database Server

Previously, Xray would synchronize with the global database server automatically at set time intervals. To give you more control over usage of your system resources, you can now manually invoke initial synchronization and update with the global database server, and pause/resume synchronization if necessary.

Support for Non-Docker Installation

In addition to Docker, JFrog Xray is now available for installation in a variety of flavors including Ubuntu, CentOS, Red Hat, and Debian.

Support Download Blocking

JFrog Xray will annotate artifacts that have been identified with an issue in any connected instance of JFrog Artifactory so that the Artifactory administrator may block download of that artifact.

Integration with Aqua

If you have an account with Aqua, this integration lets you enable their feed as a source for alerts using your Aqua API key.

OSS License Policy

You may now implement an OSS license policy by defining a filter for watches based on a whitelist or blacklist of OSS licenses. Any component in the system that does not pass through the filter you define will generate an alert.

Main Updates

1. Support for older versions of Artifactory - v4.0 and above
2. Visibility and control over resources with synchronizing with the global database server
3. Support for Linux installations
4. Support download blocking
5. Support for manually invoking and operating synchronization with the global database server
6. Integration with Aqua
7. OSS license policies
8. Connect to Artifactory via an HTTP proxy.

Xray 1.0

August 1, 2016

JFrog is proud to the first official release JFrog Xray 1.0. This version presents dramatic changes based on feedback recieved from customers using the previous "Preview" version released several weeks ago.

Easy Onboarding

The entire onboarding process to get started with Xray is done within Xray. This includes adding Artifactory instances, specifying repositories for indexing, triggering indexing and getting status on the indexing process.

Unified Analysis

Watches and alerts now aggregate all types of analysis performed. You simply define the context you are interested in for a **Watch** (repository, build or all artifacts), and view aggregated information on issues detected and artifacts impacted in the resulting alert.

Focusing on the most relevant issues and alerts

You can now choose to ignore alerts or issues that have been resolved or are not interesting to you either for a specific alert instance or permanently.

Integrations

While JFrog Xray comes preconfigured with a database of issues and affected software artifacts, it is also open to integration with additional vulnerability providers. This version comes with the ability to add Whitesource, a simple but powerful open source security and license management solution.

Manually Invoking a Scan

A new **Watch** will only apply to new Artifacts or issues that arise after it has been created. This version adds the ability to run an analysis manually and apply a new **Watch** on existing artiafcts and issues.

Main Updates

1. Easy onboarding
2. Unified analysis with Watches.
3. Focusing on important issues using "ignore" rules.
4. Integration with Whitesource.
5. Manually invoking a scan.
6. View all alerts or only those based on watches you defined.
7. Support for an HTTP proxy to communicate with external networks.

Xray 1.0.2

August 11, 2016

This is a minor update that fixes an issue with indexing and adds a limitation on the storage Xray consumes.

Main Updates

1. Fixed an issue that caused the indexing process to be terminated in certain cases.
2. Xray now limits the storage it utilizes when downloading artifacts for indexing.

Xray Preview

July 3, 2016

JFrog is proud to release JFrog Xray!

JFrog Xray performs universal artifact analysis, recursively scanning all layers of your binary packages to provide radical transparency and unparalleled insight into your software architecture. JFrog Xray works with most package formats and is fully integrated with JFrog Artifactory.

Home

The Home screen is your dashboard where you can monitor Artifactory instance Xray are connected to, component graphs and alerts.

Watches

Watches monitor artifacts for issues, and trigger alerts if any are found. A **Scanning** watch monitors a named build or repository in Artifactory and triggers an alert if any dependency with issues is found. An **Impact Analysis** watch listens to all providers streaming information to Xray and performs an impact analysis on all components in its database for any issues reported.

Alerts

Alerts provide details about any issue found with any component, showing the full infection path through the component hierarchy.

Components

View component relationships in your repositories to understand how one component affects others.

REST API

Automate component analysis through the rich Xray REST API.