# Use cases

## Description

This document describes the use cases envisioned for FOSSology-Ninka. Use cases follow the format of SPDX document use cases.[[1]](#footnote-1)

## Changelog

|  |  |  |
| --- | --- | --- |
| Date: | Action | Who |
| 02/24/2014 | Began first draft of installation use case | Jon von Kampen |
| 02/25/2014 | Completed first drafts of installation, command line scan, and web service scan use cases | Jon von Kampen |

## System administrator installs FOSSology-Ninka

1. **Title:** System administrator installs FOSSology-Ninka
2. **Primary Actor:** System administrator on behalf of users who determine artifact licenses
3. **Goal in Context:** To speed the determination of artifact licenses by comparing the output of multiple automatic license scanners
4. **Stakeholders and Interests:**
   1. **System administrators:**
      1. To provide users with the tools they need for their tasks
      2. To ensure the availability and responsiveness of all user applications
   2. **License determiners:**
      1. To determine and communicate artifact licenses
      2. To automate and quicken the license determination process
5. **Preconditions:**
   1. Installation of FOSSology and Ninka with appropriate system resources to run them serially
   2. Installation of FOSSology-Ninka infrastructure components (e.g., Python interpreter)
6. **Main Success Scenario:**
   1. FOSSology-Ninka can successfully locate and execute FOSSology and Ninka
   2. FOSSology-Ninka is executable by all authorized users
7. **Failed End Conditions:**
   1. System administrator cannot determine how to configure FOSSology-Ninka to access its dependencies (FOSSology, Ninka, Python, etc.)
   2. FOSSology-Ninka is not executable by authorized users
8. **Trigger:** License determiner requests installation
9. **Notes:** This use case assumes that system administrators possess the requisite knowledge to install and configure FOSSology, Ninka, and Python for standalone operation (i.e., to function as expected in use cases other than FOSSology-Ninka).

# License determiner performs local FOSSology-Ninka scan through command line

1. **Title:** License determiner performs local FOSSology-Ninka scan through command line
2. **Primary Actor:** User tasked with determining artifact licenses
3. **Goal in Context:** To speed the determination of artifact licenses by comparing the output of multiple automatic license scanners
4. **Stakeholders and Interests:**
   1. **License determiners:**
      1. To determine and communicate artifact licenses
      2. To automate and quicken the license determination process
   2. **Artifact consumers:**
      1. To receive accurate and clear artifact licensing information
      2. To be able to comply easily with artifact licenses
5. **Preconditions:**
   1. Artifacts contain some licensing information (e.g., in comments)
   2. FOSSology and/or Ninka accurately assert licensing information
   3. The license determiner has the knowledge, experience, and resources to manually resolve conflicting license scanner assertions
6. **Main Success Scenario:** Output allows the license determiner to accurately and completely determine artifact licenses
   1. A provisional SPDX document is output to the local file system
   2. SPDX fields on which FOSSology and Ninka agree are filled out
   3. SPDX fields on which FOSSology and Ninka conflict are marked NO ASSERTION
   4. A data structure within the SPDX Comments field identifies conflict fields and lists the conflicting assertions
7. **Failed End Condition:**
   1. Failure to run FOSSology or Ninka on a local package, or
   2. Failure to compare FOSSology and Ninka output, or
   3. Failure to create a provisional SPDX document on the local file system
8. **Triggers:**
   1. User is tasked with determining licenses for artifacts belonging to the user’s organization
   2. User is tasked with determining licenses for third-party artifacts that the user’s organization would like to use
   3. Manual or automated command-line execution of FOSSology-Ninka
9. **Notes:** FOSSology-Ninka itself does not guarantee accurate and complete determination of artifact licenses. Artifacts may lack (complete) licensing information; FOSSology or Ninka determinations may be inaccurate; the license determiner may be unable to manually resolve conflicting license scanner assertions. FOSSology-Ninka aims to quicken the license determination process over purely manual review.
10. **Example:** A user is tasked with determining the licenses used in a third-party software package. The user provides the local path to the package as an argument to FOSSology-Ninka. FOSSology-Ninka calls FOSSology and Ninka serially on the package, compares their assertions, and outputs a provisional SPDX document. The document records that both scanners assert that the license is GPLv3. The user may use some other tool to create a final SPDX document indicating human review.

## License determiner performs remote FOSSology-Ninka scan through web service call

1. **Title:** License determiner performs remote FOSSology-Ninka scan through web service call
2. **Primary Actor:** User tasked with determining artifact licenses
3. **Goal in Context:** To speed the determination of artifact licenses by comparing the output of multiple automatic license scanners
4. **Stakeholders and Interests:**
   1. **License determiners (including SPDX Dashboard users):**
      1. To determine and communicate artifact licenses
      2. To automate and quicken the license determination process
   2. **Artifact consumers:**
      1. To receive accurate and clear artifact licensing information
      2. To be able to comply easily with artifact licenses
5. **Preconditions:**
   1. Artifacts contain some licensing information (e.g., in comments)
   2. FOSSology and/or Ninka accurately assert licensing information
   3. The license determiner has the knowledge, experience, and resources to manually resolve conflicting license scanner assertions
6. **Main Success Scenario:** Output allows the license determiner to accurately and completely determine artifact licenses
   1. A provisional SPDX document is uploaded to an instance of the SPDX Dashboard web application
   2. SPDX fields on which FOSSology and Ninka agree are filled out
   3. SPDX fields on which FOSSology and Ninka conflict are marked NO ASSERTION
   4. A data structure within the SPDX Comments field identifies conflict fields and lists the conflicting assertions
7. **Failed End Condition:**
   1. Failure to run FOSSology or Ninka on a package downloaded from a remote location, or
   2. Failure to compare FOSSology and Ninka output, or
   3. Failure to create a provisional SPDX document and upload it to the SPDX Dashboard
8. **Triggers:**
   1. User is tasked with determining licenses for artifacts belonging to the user’s organization
   2. User is tasked with determining licenses for third-party artifacts that the user’s organization would like to use
   3. Manual or automated call to a web service that executes FOSSology-Ninka
9. **Notes:** FOSSology-Ninka itself does not guarantee accurate and complete determination of artifact licenses. Artifacts may lack (complete) licensing information; FOSSology or Ninka determinations may be inaccurate; the license determiner may be unable to manually resolve conflicting license scanner assertions. FOSSology-Ninka aims to quicken the license determination process over purely manual review.
10. **Example:** A user is tasked with determining the licenses used in a third-party software package. The user (through a web front-end) calls a FOSSology-Ninka web service with the path to a remote package as an argument. FOSSology-Ninka downloads the package to the local file system. FOSSology-Ninka calls FOSSology and Ninka serially on the package, compares their assertions, and outputs a provisional SPDX document. The document records that both scanners assert that the license is GPLv3. The document is uploaded to the SPDX Dashboard where the user may create a final SPDX document indicating human review.

1. <http://wiki.spdx.org/view/Technical_Team/Use_Cases/2.0> [↑](#footnote-ref-1)