Data Flow Diagram

# Description

This document describes the data flow between components of FOSSology-Ninka and third-party components.

# Changelog

|  |  |  |
| --- | --- | --- |
| Date: | Action | Who |
| 01/27/2014 | Created Prototype | Doug Richardson |
| 01/31/2014 | Created MS-Paint Prototype of the dataflow diagram | Doug Richardson |
| 01/31/2014 | Created VISIO document of the dataflow diagram | James Thompson |
| 02/04/2014 | Moved into final report | Jon von Kampen |
| 02/05/2014 | Updated to reflect comments during class presentation | James Thompson |
| 02/26/2014 | Updated DFD and inserted pseudo code into DFD doc, also inserted comments in pseudo code. | James Thompson |
| 02/26/2014 | Added document description  Proofread and reformatted annotations | Jon von Kampen |

# Data Flow Diagram



# DFD Annotations

## Run Scanners

### Description

Run Scanners will collect the file or package from the end-user and send it to both FOSSology and Ninka. Both tools attempt to automatically locate license declarations within files. Both tools are external modules and treated as black boxes for the purpose of this diagram.

### Mock-up Code

<https://github.com/TheFinks/Fossology-Ninka/tree/master/code_mockups/run_scanner>

### Procedure

1. INPUT (FILE or PACKAGE)
2. IF INPUT is FILE:
   1. FOSSology (FILE)
   2. Ninka (FILE)
3. ELSE:
   1. UNPACK PACKAGE as TEMP
   2. FOR EACH FILE in TEMP:
      1. FOSSology (FILE)
      2. Ninka (FILE)

## Combine License Docs

### Description

Combine License Docs will be used to accept the output of both the FOSSology and Ninka scanning tools. After accepting both outputs this module will compare and combine the outputs.

### Procedure

1. INPUT (FOSSOLOGY\_OUT, NINKA\_OUT)
2. READ (FOSSOLOGY\_OUT) as R1
3. READ (NINKA\_OUT) as R2
4. COMPARE (R1, R2): *# Compares R1 and R2 line by line; assumes the files are sorted in the same order*
   1. IF R1 has LICENSE and R2 has NONE or ERROR:
      1. WRITE R1.LICENSE to COMBINED\_INFO
   2. ELSE IF R2 has LICENSE and R1 has NONE or ERROR:
      1. WRITE R2.LICENSE to COMBINED\_INFO
   3. ELSE IF R1.LICENSE == R2.LICENSE:
      1. WRITE R1.LICENSE to COMBINED\_INFO *# R1 and R2 are identical; either one works*
   4. ELSE IF R1.LICENSE != R2.LICENSE:
      1. WRITE “LICENSE\_DECLARED = NOASSERTION; COMMENTS = ‘CONFLICT’” to COMBINED\_INFO *# Not the actual output format; just an example*

## SPDX Generator

### Description

The SPDX generator will accept the output from the Combine License Docs module and convert the document into a JSON SPDX file.

### Mock-up Code

<https://github.com/TheFinks/Fossology-Ninka/tree/master/code_mockups/spdx_generator>

### Procedure

1. INPUT (COMBINED\_INFO) *# Combined license documentation*
2. CREATE FILE (NAME = FINAL\_DOC, FORMAT = JSON, SCHEMA = SPDX-1.2)
3. IF COMBINED\_INFO.FILE\_FORMAT == PACKAGE:
   1. FOR LINE in COMBINED\_INFO: *# Multiple lines for a package*
      1. WRITE (LINE.LICENSE\_DECLARED) to FINAL\_DOC.LICENSE\_DECLARED
      2. WRITE (LINE.COMMENTS) to FINAL\_DOC.COMMENTS *# Even if no conflicts are recorded in the comments, this ensures comments for other purposes are included*
4. ELSE:
   1. WRITE (LINE.LICENSE\_DECLARED) to FINAL\_DOC.LICENSE\_DECLARED
   2. WRITE (LINE.COMMENTS) to FINAL\_DOC.COMMENTS