PMR Insight Collective Knowledge (PICK)

Software Design Document

Version 1.9.2

2/27/2020

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 1.9.2 |
| Indicator of Last Page in Document: | $$$ |
| Date of Last Review: | 3/7/2020 |
| Date of Next Review: | 3/8/2020 |
| Target Date for Next Update: | 3/8/2020 |

Distribution List

This following list of people shall receive a copy of this document every time a new version of this document becomes available:

Guidance Team Members: Dr. Steven Roach

Jake Lasley

Customer: Mr. Baltazar Santaella

Ms. Herandy Vasquez

Mr. Vincent Fonseca

Ms. Florencia Larsen

Mr. Eric de Nava

Software Team Members: Itzel Rivas

Eder Rodriguez

Jose Gallardo

Gilbert Alvarez

Nusrat Sarmin

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 2/27/2020 | Gilbert Alvarez | Creation of Document |
| 1.1 | 2/28/2020 | Gilbert Alvarez | Purpose and Intended Audience |
| 1.1.1 | 2/28/2020 | Itzel Rivas | Scope and helped with purpose |
| 1.2 | 2/29/2020 | Jose Gallardo | Definitions, Acronyms, Abreviation |
| 1.2.1 | 2/29/2020 | Nusrat Sarmin | Definitions |
| 1.3 | 3/4/2020 | Gilbert Alvarez | Over view and V1 of Wriffs-Brock diagram. |
| 1.3.1 | 3/4/2020 | Eder Rodriguez | Modified Wriffs-Brock diagram |
| 1.4 | 3/5/2020 | Gilbert, Eder, Itzel, Jose, Nusrat | Worked on improving CRC |
| 1.4.1 | 3/5/2020 | Gilbert Alvarez | Subsystem and comment descriptions |
| 1.5 | 3/5/2020 | Jose Gallardo | Detailed Description of components (user subsystem) |
| 1.5.1 | 3/5/20 | Eder Rodriguez | Edited System Component descriptions |
| 1.6 | 3/6/2020 | Eder Rodriguez | Described Dependencies |
| 1.7 | 3/7/2020 | Gilbert Alvarez | Modified Wriffs-Brock diagram and added description |
| 1.8 | 3/7/2020 | Itzel Rivas | 3.3 User interface Subsystem |
| 1.8.1 | 3/7/2020 | Itzel Rivas | 3.4 Ingestion/Validation/Cleansing Subsystem |
| 1.8.2 | 3/7/2020 | Nusrat Sarmin | Contract 6 |
| 1.9 | 3/8/2020 | Jose Gallardo | Contracts for Ingestion/Validation/Cleansing subsystem |
| 1.9.1 | 3/8/2020 | Gilbert Alvarez | Modify diagram again to have appropriate contract numbers, fixed contracts of classes in section 3 |
| 1.9.2 | 3/8/2020 | Gilbert Alvarez | Created Database Schema for VectorDB |

Table of Contents

[Document Control ii](#_Toc34561161)

[Approval ii](#_Toc34561162)

[Document Change Control ii](#_Toc34561163)

[Distribution List ii](#_Toc34561164)

[Change Summary ii](#_Toc34561165)

[1. Introduction 1](#_Toc34561166)

[1.1. Purpose and Intended Audience 1](#_Toc34561167)

[1.2. Scope of Product 1](#_Toc34561168)

[1.3. References 1](#_Toc34561169)

[1.4. Definitions, Acronyms, and Abbreviations 1](#_Toc34561170)

[1.4.1. Definitions 1](#_Toc34561171)

[1.4.2. Acronyms 2](#_Toc34561172)

[1.4.3. Abbreviations 2](#_Toc34561173)

[1.5. Overview 2](#_Toc34561174)

[2. Decomposition Description 3](#_Toc34561175)

[2.1. System Collaboration Diagram 3](#_Toc34561176)

[2.2. Subsystem and Component Descriptions 3](#_Toc34561177)

[2.2.1. Visualization Subsystem 3](#_Toc34561178)

[2.2.2. User Subsystem 4](#_Toc34561179)

[2.2.3. User Interface Subsystem 4](#_Toc34561180)

[2.2.4. Ingestion/Validation/Cleansing Subsystem 4](#_Toc34561181)

[2.3. Dependencies 4](#_Toc34561182)

[3. Detailed Description of Component 6](#_Toc34561183)

[3.1. Visualization Subsystem 6](#_Toc34561184)

[3.1.1. Node 6](#_Toc34561185)

[3.1.2. Vector 6](#_Toc34561186)

[3.1.3. Connector 7](#_Toc34561187)

[3.1.4. Q Graph Viz 7](#_Toc34561188)

[3.2. User Subsystem 8](#_Toc34561189)

[3.2.1. Analyst 8](#_Toc34561190)

[3.2.2. Lead Analyst 8](#_Toc34561191)

[3.3. User Interaction Subsystem 9](#_Toc34561192)

[3.3.1. UI View Manager 9](#_Toc34561193)

[3.3.2. UI Graph and Table view 9](#_Toc34561194)

[3.3.3. UI Vector Config View 10](#_Toc34561195)

[3.3.4. UI Action Report View 10](#_Toc34561196)

[3.3.5. UI Team Configuration 11](#_Toc34561197)

[3.4. Ingestion/Validation/Cleansing 11](#_Toc34561198)

[3.4.1. Splunk 11](#_Toc34561199)

[3.5. Log File Class 11](#_Toc34561200)

[3.6. Log Entry 12](#_Toc34561201)

[3.7. Enforcement Action Report 12](#_Toc34561202)

[3.8. Event Configuration 13](#_Toc34561203)

[3.9. Vector DB 13](#_Toc34561204)

[4. Database 14](#_Toc34561205)

[4.1. Database Schema 14](#_Toc34561206)

# Introduction

## Purpose and Intended Audience

The purpose of the Software Design Document (SDD) is to identify collaborations and subsystems within the PMR Insight Collective Knowledge (PICK) software to be able to design interactions within the system. The SDD describes what is to be built and is intended to help the development team understand what is expected to be built. Sections within the SDD include the system and subsystem collaboration graphs, component descriptions, descriptions of the classes within the components, contracts between classes, and description of database schemas. The SDD focuses on describing contracts of classes which are client server relationships that occur in order to fulfill a client’s responsibilities, and protocols of classes that state each method in a class, parameters of the methods, return value of the method, purpose of the method and and pre and post conditions of a method.

## Scope of Product

This Software Design Document focuses on the core functionality of the PICK system and its main components, which is to streamline the job of an analyst to create a report from log files submitted by the defense and attacking teams. The system will utilize outside libraries and pre-existing systems, described on section 2.3, to achieve artifact ingestion, validation, cleansing and creating a visual representation of significant events for the analyst to see. Our goal is to be able to aid analysts in the creation of a report from any form of artifacts containing information of a attack response situation.

## References

[1] 08c\_SDD\_Outline.pdf

[3] 08aContracts.pdf

[4] 08b\_Protocols.pdf

[5] 09Subsystems.pdf

[6] Hasneeza. “Example of Software Design Document (SDD).” UniMAP Portal, 26 Apr. 2012, portal.unimap.edu.my/portal/page/portal30/Lecture Notes/KEJURUTERAAN\_KOMPUTER/Semester 2 Sidang Akademik 20112012/EKT420 Software Engineering/Example of Software Design Document (SDD).

## Definitions, Acronyms, and Abbreviations

### Definitions

|  |  |
| --- | --- |
| **Lead Analyst** | The analyst responsible for the oversight of work being performed by the other analysts, some other responsibilities include delegating work, and approving work |
| **Analyst** | A user that is authorized to work PICK system, they must connect to the Lead analysts' machine to gain access to the log files and databases. Some of their responsibilities include editing, creating, deleting log entries/nodes, and pushing work to the Lead analyst for approval upon synchronization. |
| **Log File** | A file that records events from a team users' machine. |
| **Log Entry** | A file that records events from a team users' machine. |
| **Log Creator** | Node attribute that states what team created the log. |
| **Node** | Visual representation of log entry on a vector graph. |
| **Node name** | Node attribute, unique name of a node. |
| **Start Date** | Start date and time of Assessment |
| **End Date** | End date and time of Assessment |
| **Timestamp** | Time the activity occurred. Recorded in Zulu time. |
| **Node ID** | Unique numerical ID created for nodes. |

### Acronyms

|  |  |
| --- | --- |
| **PICK** | PMR Insight Collective Knowledge |
| **PMR** | Prevent, Mitigate, Recover |
| **SDD** | Software Design Document |
| **UML** | Unified Modeling Language |

### Abbreviations

|  |  |
| --- | --- |
| **DB** | Database |
| **config** | Configuration |
| **e.g.** | For example |
| **i.e.** | in other words |

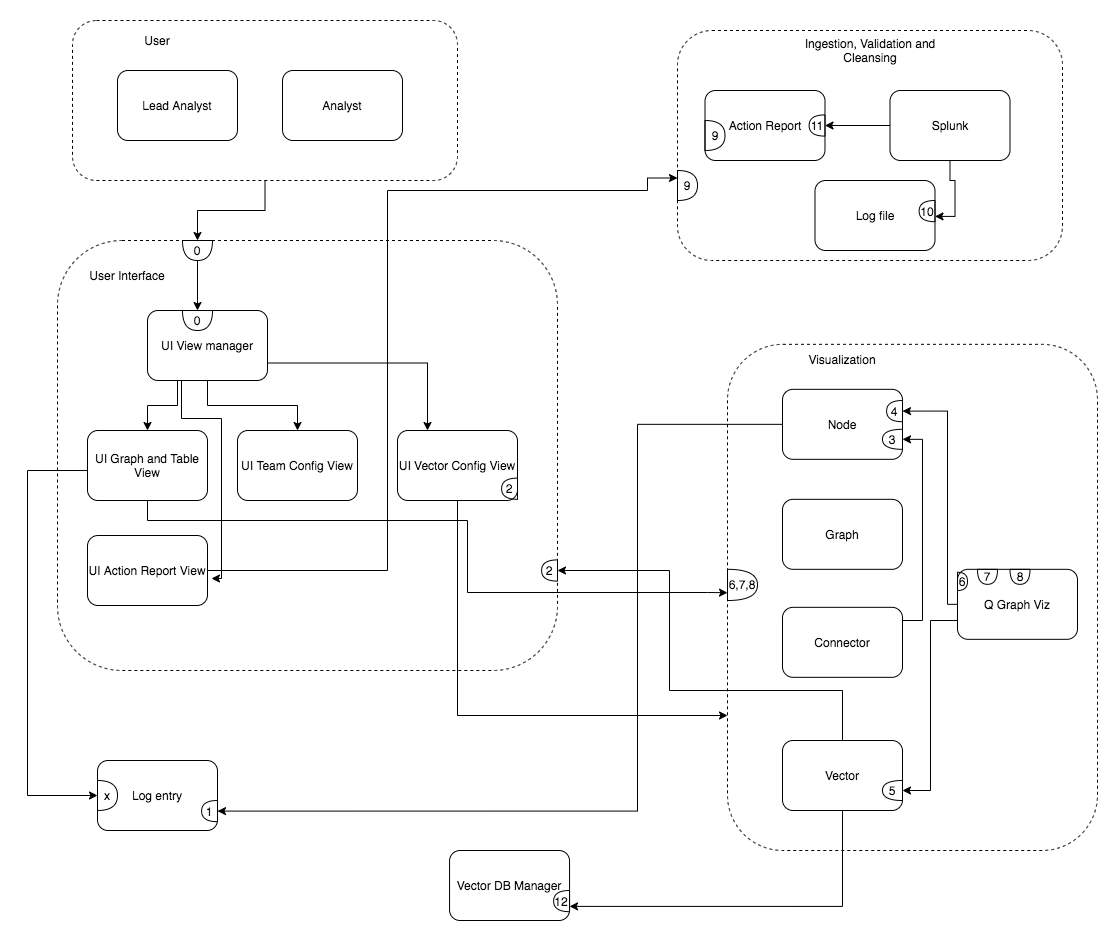
## Overview

The following section describes the components and the interactions of the subsystems in the PICK system using Wriffs-Brocks diagram. Wriffs-Brock diagram utilizes rounded squares to indicate classes and semicircles with numbers and arrows pointing from one class to another to indicate a contract between two classes. Subsystems on the Wriffs-Brock diagram are indicated with a dotted container surrounding the classes. This section also describes the design choices used for dividing design entities.

# Decomposition Description

<< Provide a description of how your component descriptions can be used by designers and maintainers identify major design entities, for purposes such as determining which entity is responsible for specific functions and tracing requirements to design entities. >>

## System Collaboration Diagram



*figure 1.0*

The system can be broken down into 4 major subsystems, User subsystem, User interface subsystem, Ingestion/Validation/Cleansing subsystem, and Visualization subsystem.

## Subsystem and Component Descriptions

### Visualization Subsystem

**Description:** This subsystem shows the interaction of the data that was created from log files coming from the ingestion/validation/cleansing process. The visualization subsystem takes the data from the log entries and draws representations of nodes and connectors using an external library, Q Graph Viz and. From there the nodes and connectors are displayed to the user on the UI view Graph and Table. This subsystem also displays log entries to the user on a table view on the same UI view.

**Purpose:** Take in data to transform it into visuals.

**Contracts:** 1 (described in table 1.0)

2 (described in table 1.1)

3 (described in table 1.2)

4 (described in table 1.3)

5 (described in table 1.3)

### 

### User Subsystem

**Description:** There are two types of users of the system, the analyst and the lead analyst. The analyst can perform changes to the DB and see everyone else who is connected to the system, but the lead analyst can approve or deny change request made to the database as well as host all connections to their machine. Lead analyst can do anything an analyst can do; however, the lead analyst has some attributes that an analyst cannot do.

**Purpose:** Describes the types of users interacting with the system and what information they provide to the system.

**Contracts:**

### User Interface Subsystem

**Description:** Then we have the User Interface subsystem that can allows the user to the interact with the system by displaying information and asking the user for input. The classes within this subsystem can display information between various views, such as the graph and table view displaying log entry and node information and the vector configuration asking the user for input to name and describe a vector.

**Purpose:** Describes the appropriate information to display to the user when they reach a certain view of the system. Asks the user for input where necessary.

**Contracts:** 6 (described in table 3.1)

7 (described in table 3.1)

8 (described in table 3.1)

### Ingestion/Validation/Cleansing Subsystem

**Description:** This subsystem describes the process of ingestion, validation and cleansing of log files. Classes associated with this subsystem include the Enforcement action report, Log files and Splunk. When a log file is ingested the result of the action of weather it led to an error or was successfully ingested is recorded into the Action Report.

**Purpose:** To show the interaction of log file and Splunk to create log entries that the system will use for the Graph and table view.

**Contracts:** 10 (described in table 4.0)

11 (described in table 4.3)

12 (described in table 4.5)

## Dependencies

The system runs using Python version 3 making the code portable with the ability to be able to run on the client's choice of operating system (Kali Linux). The Visual Representation component that contains classes such as Graph and node, drawing the graph utilizes the external library QGraphViz to draw nodes and edges making up the graph. The UI controller component uses an external library pyqt to draw user interface elements and create views such as Graph and table view, event configuration view, directory view, Action Report view, Log ingestion/validation/cleansing view, and Team configuration view. Ingestion of the log files in the system depend on an external system, Splunk.

# Detailed Description of Component

## Visualization Subsystem

### Node

<< For each class in this component, create a new Heading 2 subsection. In each subsection, describe the internal details of each design entity. Each class should have a class name, a description, super classes, and private responsibilities. >>

*Table 1.0*

|  |  |
| --- | --- |
| **Class Name**: Node | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Knows its own unique name. 2. Knows its icon type. 3. Knows its visibility. 4. Knows its unique node ID. | |
| **Contract:** 1 Get Log Entry Information | |
| **Responsibilities** | **Collaborations** |
| 1. Know Start time. 2. Know End time. 3. Know source document. 4. Know log entry description. 5. Know log creator. | Log Entry (1,2,3,4,5) |

**Signature:**

* **Method Name:** getNodeInfo();
* **Return Value Type:** void
* **Input and output Parameters:**
* **Description of Parameters:**
* **Purpose:**

**Pre-Conditions:** Log Entries must exist.

**Post-Conditions:** Node object has been declared.

### Vector

*Table 1.1*

|  |  |
| --- | --- |
| **Class Name**: Vector | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Knows its name. 2. Knows its description. | |
| **Contract:** 2 Get Name. | |
| **Responsibilities** | **Collaborations** |
| 1. get name from user input. | UI Vector Configuration (1) |

**Signature:**

* **Method Name:** getVectorName();
* **Return Value Type:** String
* **Input and output Parameters:**
* **Description of Parameters:** User entered string for a vector.
* **Purpose:**

**Pre-Conditions:** The user must enter a string name from the UI Vector Configuration view and create a vector.

**Post-Conditions:** Returns the vector name as a string.

### Connector

*Table 1.2*

|  |  |
| --- | --- |
| **Class Name**: Connector | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Knows its Relationship ID. 2. Knows its parent node ID. 3. Knows its child node ID. 4. Knows its description. | |
| **Contract:** 3 Get the IDs of the nodes being connected. | |
| **Responsibilities** | **Collaborations** |
| 1. Know what nodes are being connected. 2. Get node ID from the first node. 3. Get node ID from the next node. | Node Class (4) |

**Signature:**

* **Method Name:** getNodeID();
* **Return Value Type:** void
* **Input and output Parameters:** 2 node Objects.
* **Description of Parameters:** Nodes that have already been created and know their unique node IDs.

**Purpose:**

**Pre-Conditions:** At least 2 nodes must exist.

**Post-Conditions:** Returns a node ID of the first node and the second node.

### Q Graph Viz

*Table 1.3*

|  |  |
| --- | --- |
| **Class Name**: Q Graph Viz | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Draws nodes. 2. Draws connectors | |
| **Contract:** 4 Draw nodes on graph. | |
| **Responsibilities** | **Collaborations** |
| 1. Know how to draw a node | Node (1) |
| **Contract:** 5 Draw connectors on graph | |
| **Responsibilities** | **Collaborations** |
| 1. Know how to draw a connector | Connector (2) |

Contract 4

**Signature:**

* **Method Name:** drawNode(Node);
* **Return Value Type:** void
* **Input and output Parameters:** Node object
* **Description of Parameters:** A node must be instantiated to be able to draw it.

**Purpose:** to draw nodes to display for the user.

**Pre-Conditions:** At least one node object must exist.

**Post-Conditions:** Representation of a node has been drawn.

Contract 5

**Signature:**

* **Method Name:** drawConnector(Connetor);
* **Return Value Type:** void
* **Input and output Parameters:** Connector object
* **Description of Parameters:** A connector must be instantiated to be able to draw it.

**Purpose:** to draw connectors to display for the user.

**Pre-Conditions:** There must exist 2 nodes to draw a connection between them.

**Post-Conditions:** Representation of a connection has been drawn between two nodes.

## User Subsystem

### Analyst

*Table 2.0*

|  |  |
| --- | --- |
| **Class Name**: Analyst | |
| **Superclass**: | |
| **Subclasses**: Lead Analyst | |
| **Private Responsibilities**   1. Can describe a vector. 2. Can name a vector. 3. Can push changes to vector DB. 4. Can pull changes from vector DB. 5. Knows if they are the lead. | |
| **Contract:** | |
| **Responsibilities** | **Collaborations** |
|  |  |

### Lead Analyst

*Table 2.1*

|  |  |
| --- | --- |
| **Class Name**: Lead Analyst | |
| **Superclass**: Analyst | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Can approve changes pushed to Vector DB. 2. Can deny changes pushed to the Vector DB. | |
| **Contract:** | |
| **Responsibilities** | **Collaborations** |
|  |  |

## User Interaction Subsystem

### UI View Manager

*Table 3.0*

|  |  |
| --- | --- |
| **Class Name**: UI view manager | |
| **Superclass**: | |
| **Subclasses**: UI Graph and Table view, UI Team Config View, UI Vector Config view, UI Action Report view | |
| **Private Responsibilities**   1. Know what view to display | |
| **Contract:** | |
| **Responsibilities** | **Collaborations** |
|  |  |

### UI Graph and Table view

*Table 3.1*

|  |  |
| --- | --- |
| **Class Name**: UI Graph and Table view | |
| **Superclass**: UI View Manager | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Display information about graph. 2. Display information about table. 3. Receive user input for creating a node. 4. Receive user input for creating new log entry. | |
| **Contract:** 6 Display nodes | |
| **Responsibilities** | **Collaborations** |
| 1. Display nodes on the graph view | Q Graph Viz |
| Contract: 7 Display nodes and Connector | |
| **Responsibilities** | **Collaborations** |
| 1. Display connectors on the graph view | Q Graph Viz |
| **Contract:** 8 Export Graph | |
| **Responsibilities** | **Collaborations** |
| 1. Display export button to export graph. 2. Give user option to export graph as jpg, cvs. | Q Graph Viz |

Contract 6

**Signature:**

* **Method Name:** dispayNodeView();
* **Return Value Type:** void
* **Input and output Parameters:**
* **Description of Parameters:** A node must be drawn to be able to display it.

**Purpose:** to display nodes to display for the user.

**Pre-Conditions:** At least one node object must exist.

**Post-Conditions:** Representation of a node has been drawn on the Graph/ Table view.

Contract 7

**Signature:**

* **Method Name:** displayConnectionView();
* **Return Value Type:** void
* **Input and output Parameters:**
* **Description of Parameters:** A connector and node must be drawn to be able to display it.

**Purpose:** to draw connectors to display for the user.

**Pre-Conditions:** There must exist 2 nodes that have drawn and a connection between them.

**Post-Conditions:** Representation of a connection has been displayed between two nodes.

Contract 8

**Signature:**

* **Method Name:** exportGraph();
* **Return Value Type:** void
* **Input and output Parameters:**
* **Description of Parameters:**

**Purpose:** to allow the user to export the current graph as a jpg, png, csv.

**Pre-Conditions:** there are vectors with log entries associated to them.

**Post-Conditions: t**he user can save a jpg, png, csv.

### UI Vector Config View

*Table 3.2*

|  |  |
| --- | --- |
| **Class Name**: UI Vector Config View | |
| **Superclass**: UI View Manager | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Ask user for Vector name input 2. Ask user for Vector description input | |
| **Contract:** | |
| **Responsibilities** | **Collaborations** |
|  |  |

### UI Action Report View

*Table 3.3*

|  |  |
| --- | --- |
| **Class Name**: UI Action Report View | |
| **Superclass**: UI View Manager | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Display error message of line from log file where ingestion failed. 2. Display progress of ingestion, validation and cleansing 3. Ask the user if they want to override error. | |
| **Contract:** 9 Display Action Report | |
| **Responsibilities** | **Collaborations** |
| 1. Display progress of ingestion 2. Display error resulting from failure to ingest | Enforcement Action Report (1,2) |

Contract 9

**Signature:**

* **Method Name:** displayAction();
* **Return Value Type:** void
* **Input and output Parameters:**
* **Description of Parameters:**

**Purpose:** to allow the user to see errors and progress of ingestion.

**Pre-Conditions:** log files are being ingested.

**Post-Conditions:** the user can see all the files that were processed.

### UI Team Configuration

*Table 3.4*

|  |  |
| --- | --- |
| **Class Name**: Team Configuration View | |
| **Superclass**: UI View Manager | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Ask analyst if they are the lead. 2. Ask Lead analyst for IP address. | |
| **Contract:** | |
| **Responsibilities** | **Collaborations** |
|  |  |

## Ingestion/Validation/Cleansing

### Splunk

*Table 4.0*

|  |  |
| --- | --- |
| **Class Name**: Splunk | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Can Ingest Log files | |
| **Contract:** 10 ingest log files | |
| **Responsibilities** | **Collaborations** |
| 1. Ingest 2. Know what to cleanse 3. Know where to retrieve log files | Log file |

Contract 10

**Signature:**

* **Method Name:** getLogfile();
* **Return Value Type:** boolean
* **Input and output Parameters:**
* **Description of Parameters:** true if the logfile was processed successfully, false if it failed.

**Purpose:** to ingest/validate/cleanse log files to turn them into log entry.

**Pre-Conditions:** Log files are stored at the directory location.

**Post-Conditions:** Log files have been ingested or resulted in an error.

## Log File Class

*Table 4.1*

|  |  |
| --- | --- |
| **Class Name**: Log File | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Knows its name. 2. Knows its cleansing status. 3. Knows its ingestion status. 4. Knows its validation status. | |
| **Contract:** | |
| **Responsibilities** | **Collaborations** |
|  |  |

## Log Entry

*Table 4.2*

|  |  |
| --- | --- |
| **Class Name**: Log Entry | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Knows start time. 2. Knows end time. 3. Knows its source document. 4. Knows a description of its activity. 5. Knows its creator. 6. Knows its log entry number. | |
| **Contract:** | |
| **Responsibilities** | **Collaborations** |
|  |  |

## Enforcement Action Report

*Table 4.3*

|  |  |
| --- | --- |
| **Class Name**: Enforcement Action Report | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Knows the line of log file that caused and error. | |
| **Contract:** 11 Know what errors occurred while ingesting | |
| **Responsibilities** | **Collaborations** |
| 1. Know what error occurred 2. Know what line the error occurred on | Slunk (1,2) |

Contract 11

**Signature:**

* **Method Name:** getStatus();
* **Return Value Type:** int: log file line
* **Input and output Parameters:**
* **Description of Parameters:**

**Purpose:** collect the log files that have not been ingested or validated because it resulted in an error.

**Pre-Conditions:** There must be log files in the directory. Splunk must have ingested a logfile.

**Post-Conditions:** Line to where the occurred is displayed on the Action report view.

## Event Configuration

*Table 4.4*

|  |  |
| --- | --- |
| **Class Name**: Event Configuration | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Knows the lead analysts IP Address. 2. Knows who the lead analyst is. 3. Knows the amount of connections allowed to the host. 4. Knows the location of the root directory. | |
| **Contract:** | |
| **Responsibilities** | **Collaborations** |
|  |  |

## Vector DB

*Table 4.5*

|  |  |
| --- | --- |
| **Class Name**: Vector DB | |
| **Superclass**: | |
| **Subclasses**: | |
| **Private Responsibilities**   1. Knows the lead analysts IP Address. 2. Knows who the lead analyst is. 3. Knows the amount of connections allowed to the host. 4. Knows the location of the root directory. | |
| **Contract:** 12 Store Vector information | |
| **Responsibilities** | **Collaborations** |
| 1. Get Vector name 2. Get Vector description 3. Store information | Vector(1,2) |

Contract 12

**Signature:**

* **Method Name:** passVectorInfo(Vector obj);
* **Return Value Type:** void
* **Input and output Parameters:** vector object
* **Description of Parameters:** vector object knows its own name and description; you can ask it for the information

**Purpose:** to store vector information on the vector DB.

**Pre-Conditions:** at least one vector has been created by or approved by analyst.

**Post-Conditions:** the vector information has been stored in the database.

# Database

## Database Schema

