PICK

Software Design Document

2.0

3/30/2020

**Document Control**

**Approval**

The Guidance Team and the customer shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 2.0 |
| Indicator of Last Page in Document: | $ |
| Date of Last Review: | 3/30/2020 |
| Date of Next Review: | 4/7/2020 |
| Target Date for Next Update: | 4/7/2020 |

**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, Clients, Software Team.

Guidance Team Members:

Dr. Ann Gates

Dr. Salamah Salamah

Dr. Steven Roach

Ms. Elsa Tai Ramirez

Customer:

Dr. Oscar Perez

Mr. Vincent Fonseca

Ms. Herandy Vazquez

Mr. Baltazar Santaella

Ms. Florencia Larsen

Mr. Erick De Nava

Software Team Members:

Mr. Valente Arellano

Mr. Tarek Haddad

Mr. Joseph Warren

Mr. Mark Williams

Mr. Ivan Torres

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |
| --- | --- | --- |
| Version | Modifier | Description |
| 1.0 | Mark Williams | Creation of Document |
| 2.0 | All Team Members | Finished document |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[**1. Introduction**](#_heading=h.c65hyhesob1p) **1**

[1.1 Intended Audience](#_heading=h.iy0flqykehfb) 1

[1.2 References](#_heading=h.lalo9gtg79q8) 1

[1.3 Definitions, Acronyms, and Abbreviations](#_heading=h.59xh4193zcz7) 1

[1.3.1 Definitions](#_heading=h.3rdcrjn) 1

[1.3.2 Acronyms](#_heading=h.90driyhzvnls) 2

[1.3.3 Abbreviations](#_heading=h.gquhacx3ozaj) 3

[1.4 Overview](#_heading=h.1ksv4uv) 3

[**2. Decomposition Description**](#_heading=h.41awc6o0kzc3) **3**

[2.1 System Collaboration Diagram](#_heading=h.sism2oh04m1u) 3

[2.2 Subsystem and Component Descriptions](#_heading=h.2veb6mtr3awz) 4

[2.3 Dependencies](#_heading=h.m3c2jkhjzfrj) 8

[**3. Detailed Description of Log Entry Component**](#_heading=h.wdo8fpj1b82y) **8**

[3.1 Log Entry](#_heading=h.50eseeqq1eg9) 8

[3.1.1.Retreive Log Entry Information](#_heading=h.1mws4y3n809o) 8

[3.2 Log Entry Configuration](#_heading=h.3zvnpf7cw30n) 10

[3.3. Log Entry Manager](#_heading=h.o1udmucph9hf) 10

[3.3.1. Retrieve Log Entries](#_heading=h.al496d8uexc9) 11

[3.3.2. Associate Log Entry to Vector](#_heading=h.plpwjqfnqhoq) 11

[**4. Detailed Description of Event Configuration Component**](#_heading=h.u4b3al5elpgu) **12**

[4.1 Event Configuration](#_heading=h.rj9osq651qyr) 12

[4.1.1.Retreive Event Configuration Information](#_heading=h.c1cmn3mcj3nl) 13

[4.2 Event Configuration Window](#_heading=h.al7zvb7fcpdt) 15

[**5. Detailed Description of Vector Db Component**](#_heading=h.l73q6sgvahsa) **15**

[5.1 Vector Db](#_heading=h.6burw6o7ua21) 16

[5.1.1.Retreive Vectors](#_heading=h.9ans3fs98otf) 16

[5.2. Vector Db Configuration](#_heading=h.ot68sp2ifjmh) 16

[**6. Detailed Description of Client/Server Component**](#_heading=h.9rdtw7lnweu0) **17**

[6.1 Client Handler](#_heading=h.ix4yanv7wwgu) 17

[6.1.1. Send Store Data Request](#_heading=h.l63iuqjbdfwu) 17

[6.1.2. Send Get Data Request](#_heading=h.wxtpoc64ogf8) 18

[6.2. Server Handler](#_heading=h.62g1i5q84c0f) 18

[6.2.1. Handle Store Data Request](#_heading=h.wajnki6b6rhq) 19

[6.2.2. Handle Get Data Request](#_heading=h.mdk7fs5c85yz) 19

[**7. Detailed Description of Log File Component**](#_heading=h.4lrl3wywj1hu) **20**

[7.1 Log File](#_heading=h.p62s9vd0vkk4) 20

[7.1.1. Validate Log File](#_heading=h.mwzzjexlmk49) 20

[7.1.2. Ingest Log File](#_heading=h.l0p9yqf5rmf) 21

[7.2 Enforcement Action Report](#_heading=h.xljdgxl7lpd7) 21

[7.3 Enforcement Action Report Manager](#_heading=h.cd75z2i43k0x) 21

[7.3.1. Retrieve Enforcement Action Report](#_heading=h.a6m8z1bwcoz5) 22

[7.4 Log File Manager](#_heading=h.sedvsew9o2yi) 22

[7.4.1. Retrieve Log Files](#_heading=h.521k7zbgx8ms) 22

[7.4.2. Modify Log File](#_heading=h.x2it5r5topbz) 23

[7.5. Log File Configuration](#_heading=h.45q8hqsz344x) 23

[**8. Detailed Description of Icon Component**](#_heading=h.t5esyddwaxq6) **23**

[8.1 Icon](#_heading=h.v5btsqenidzs) 24

[8.1.1. Retrieve Icon Image](#_heading=h.c3u2boqrcf5h) 24

[8.2 Icon Manager](#_heading=h.vhg0gjaqr4r2) 25

[8.2.1. Retrieve Icons](#_heading=h.7x2uvxj4hhi6) 25

[8.3. Icon Configuration](#_heading=h.btfev7i4l5w) 26

[**9. Detailed Description of Vector Component**](#_heading=h.l1y256feuyyp) **26**

[9.1 Vector](#_heading=h.i0r9habgc952) 26

[9.1.1. Retrieve Vector Information](#_heading=h.n5xxiv7gei83) 27

[9.1.2. Modify Node Information](#_heading=h.dfngavtz6l) 28

[9.1.3. Create New Relationship](#_heading=h.8rs15ochk3) 28

[9.2. Graph](#_heading=h.u9c7wt7648f7) 29

[9.2.1. Retrieve Graph Information](#_heading=h.4o4xhvtw4w7r) 29

[9.3. Vector Table Configuration](#_heading=h.nu0w8hxa4vuy) 29

[9.4. Graph Configuration](#_heading=h.wk6576pec9cb) 29

[9.5. Node](#_heading=h.7aqqtcycs0kj) 30

[9.5.1. Retrieve Node ID](#_heading=h.27h9l6hku5ak) 30

[**10. Detailed Description of Splunk Interface Component**](#_heading=h.vo4qtfah9rkc) **30**

[10.1. Splunk Interface](#_heading=h.oqgixyl88o6a) 30

[10.1.1. Retrieve Log Entries From Splunk](#_heading=h.d7iqgblku6yx) 31

[**11. Detailed Description of OCR Tool Interface Component**](#_heading=h.pr0zu3ni590a) **32**

[11.1. OCR Tool Interface](#_heading=h.bjvwjuhtcan) 32

[11.1. Retrieve Log Entries From the OCR Tool](#_heading=h.aiiw2d3efecn) 32

[**12. Detailed Description of Transcription Tool Interface Component**](#_heading=h.eco4sz3t5oir) **33**

[12.1. Transcription Tool Interface](#_heading=h.u8jdeki9srym) 33

[12.1. Retrieve Log Entries From the Transcription Tool](#_heading=h.sbah1086qb3s) 34

# 1. Introduction

The purpose of the Software Design Document (SDD) is to provide the required documentation and details needed to successfully build the system in development - the Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK) [1]. It will show and describe the protocols used between contracts stated in the collaboration diagram in Section 2.

## 1.1 Intended Audience

The intended audience of the SDD is Dr. Oscar Perez, Mr. Vincent Fonseca, Ms. Herandy Vazquez, Mr. Baltazar Santaella, Ms. Florencia Larsen, Mr. Erick De Nava and the software development team, Oroware (refer to Document Control, Distribution List, Software Team).

## 1.2 References

[1]  Lethality Survivability & Human Systems Integration Directorate (LSH) (2019). PMR Tool RDD V3.

[2] Santaella, B., Fonseca, V., De Nava, E. and Larsen, F. (2019). PMR Insight Collective Knowledge (PICK) Presentation.

[3]  Lethality Survivability & Human Systems Integration Directorate (LSH) (2019). PICK Needs Updated

## 1.3 Scope of Product

In this section, we review the scope of the product being developed with an emphasis on the goals of the product. First, we explore the background for the development of this product.

The product that will be produced is called the PMR (Prevent, Mitigate, Recover) Insight Collective Knowledge (PICK) Tool. We are producing this product for the LSH (Lethality Survivability & Human Systems Integration Directorate) who have specified the need for a tool that will make it easier for LSH analysts to view, correlate and report on actual cyber effects on a system that occur during an Adversarial Assessment (AA). Specifically, the clients have expressed the need to determine the events of significance that occur in an AA as well as visually represent what occurred in an AA [1].

The software product we will create will aid LSH analysts in this effort by allowing analysts to associate significant events that occur in an AA (Adversarial Assessment) to vectors. In the context of the system, these vectors will represent a series of activities/steps an adversary executes or attempts to execute that are necessary to achieve an objective [2]. Vectors will be represented in two forms in the system: a table form and a graphical form. Both representations will be able to be edited which will help analysts dynamically construct vectors. The PICK tool will also support the filtering and sorting of events that occur during an AA which will fulfill another of the analysts needs: the ability to easily find events of significance in an AA.

Essentially, the main goals of the software being produced are as follows:

1. The PICK tool will allow analysts to define vectors in the event config.
2. The PICK tool will enable analysts to discover significant events by allowing analysts to filter and sort through all of the events that occur in an AA.
3. The PICK tool will allow analysts to associate significant events to vectors which can be represented as a visual graph or table [3].
4. The PICK tool will allow analysts to correlate events into vectors.
5. The PICK tool will allow analysts to export vectors as an image of the visual graph or as a CSV file of the table.

It is important to note that the current process LSH analysts use to analyze AAs is unable to fulfill the goals our software system will provide.

## 1.4 Definitions, Acronyms, and Abbreviations

### 1.4.1 Definitions

This section lists the common terms used in this document and their associated definitions.

**Table 1: Definitions**

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| Adversarial Assessment/ Assessment | The entire process that the LSH analysts go through to evaluate the blue team’s ability to prevent, mitigate, and recover from attacks by the red team. This is done by analyzing log entries about the red team’s attacks and the blue team’s responsiveness to those attacks [2]. |
| Association | The connection between a particular log entry and a vector. |
| Correlation | The parent-child connection between two nodes of a particular vector. Two nodes that have a correlation usually mean that one action, represented by the parent node, caused some other action to occur, represented by the child. |
| Cleansing | The process of removing unwanted characters and empty rows and lines from the log files. |
| Connector | The visual representation of a correlation between two nodes of a vector that is shown by a single headed arrow pointing from parent to child. |
| Event Config | A component of the system to be produced that will allow the analysts to define elements of the assessment, including assessment name, description, start time and date, end time and date, root path, and all vectors. An assessment can not begin until all elements of the event config are defined. |
| Enforcement Action Report | A report that documents all the inaccurate data in a log file. |
| Filtering Criteria | Criteria an analyst enters that describes desired log type, time interval, and log data content. |
| Tmux log | Tmux logs are logs from a terminal multiplexer for Unix-like operating systems [3]. |
| Graph | The visual representation of a single given vector and its related nodes and connectors. |
| Ingest | The process of putting data from log files into a more readable and searchable database within the system. Files must first be validated before creating log entries from the log files. |
| Log Entry | A record of an action made by a member of the red, white, or blue team. Contains info such as: date and time of occurence, team, event type, and source log. Log entries are the result of a log file being validated and ingested into the database within the system. |
| Node | The visual representation of a particular significant log entry on the graph. Shown with an icon that should represent the type of event and is correlated with other nodes with connectors. |
| Raw/Source Log File | The log files that can be found in the directory structure located by the root path. These are ingested into the system but can not be viewed or changed directly via the system. |
| Root Path | An element of an assessment that is defined in the event config. Contains the directory path to the directory that will contain red, white, and blue subdirectories. These subdirectories then contain all the source log files. |
| Significant log entry | The abstract term for a log entry that has been associated with a particular vector. |
| Search Criteria | Criteria an analyst enters that may contain a logical search, regex, or keywords. |
| Validate | The process of reading an entire log file to ensure that every potential log entry contains all of the key attributes that makes up a log entry. |
| Vector | Created in the event config with just a name and a description, but after the analyst associates log entries to the vector and correlates nodes together, the collection of log entries in a vector can be used to describe some adversarial event or story between the blue and red team. |

### 1.4.2 Acronyms

**Table 2: Acronyms**

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| PMR | Prevent, Mitigate, Recover |
| PICK | PMR Insight Collective Knowledge |
| SRS | Software Requirements Specification |
| AA | Adversarial Assessment |
| LSH | Lethality Survivability & Human Systems Integration Directorate |
| OCR | Optical Character Recognition |
| ETL | Extract, Transform, Load |
| CSV | Comma-Separated Values |
| GUI | Graphical User Interface |
| PDF | Portable Document Format |
| DB | Database |
| PNG | Portable Network Graphics |

### 1.4.3 Abbreviations

**Table 3: Abbreviations**

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| e.g. | For example |
| i.e. | In other words |
| config. | Configuration |

## 1.5 Overview

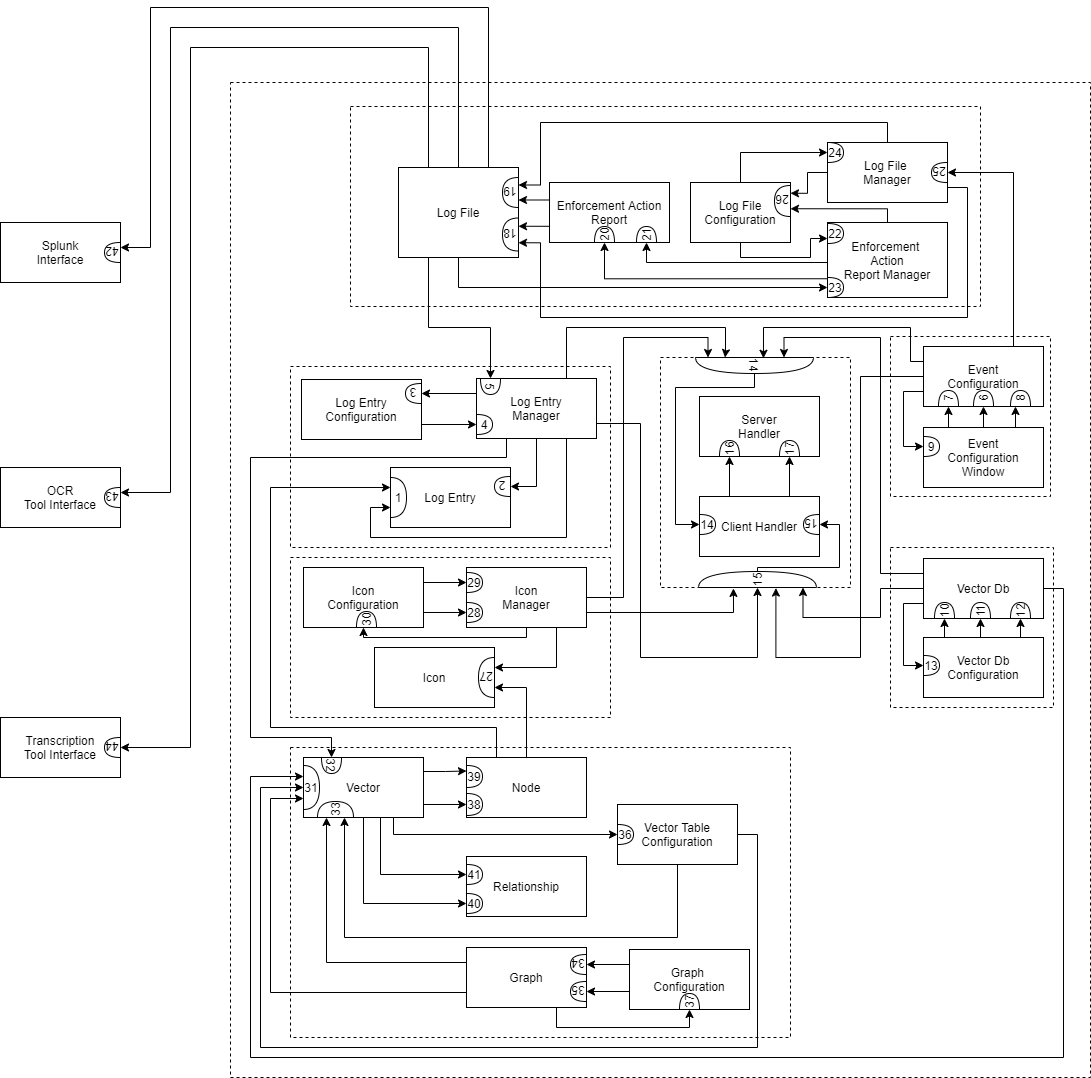
The following section, section 2.0, describes the abstract components of our system, where each class of our system is separated into these components that handle some functionality of our PICK System. We use a system collaboration diagram to visualize these separate components and all the classes within, and describe the significance of this diagram. As we will expand on in section 2.1, there exists contracts wherever different classes must interact with each other. Section 2.2 summarizes the contracts that can be found within each component of the PICK system, and which section of the SDD they are further expanded in.

The goal of the remainder of the SDD is to detail every aspect of these contracts by pairing responsibilities that classes of our system must perform for other classes and the method that gets the job done. It is important to note that not all classes collaborate with other classes to achieve a responsibility, and therefore a class may not have a contract. In which case, their private responsibilities are listed and nothing more. Otherwise, classes will contain their private responsibilities, the public responsibilities that are fulfilled with a contract with another class, and are then followed by a contract. Each contract contains a name, id, description, and a protocol. The protocol itself is a list of the public responsibilities, each containing the method signature of the method that fulfills the responsibility, the pre/post-condition of the method, and the general purpose of the method.

# 2. Decomposition Description

In this section, we describe the system being developed by reviewing each of the system’s subsystems at a high-level. We provide a collaboration diagram as well that describes the various contracts in our system. We also describe some of the impacts of our system design in this section.

## 2.1 System Collaboration Diagram



Description: The system has been structured using several highly cohesive subsystems. Each subsystem is separated so that it focuses on a key element of the system. For example, the Icon Configuration Subsystem (the subsystem containing Icon, Icon Manager, and Icon Configuration) focuses on displaying, modifying, and viewing icons. The other subsystems shown operate at a similar level of abstraction. In fact, each subsystem has three main components: a component to display data, a model of the data itself, and a high-level controller. Essentially, each subsystem implements MVC (to some extent). All subsystems interact with the Client/Server subsystem because that subsystem is in charge of storing and retrieving data from system datastores. This means that each client machine has access to all the other client machines’ information (to some extent). It is important to note that the system as a whole must interact with some external entities (Splunk, the OCR Tool, the Transcription Tool) to fulfill its responsibilities. Because of this, specific interfaces (Splunk interface, OCR Tool interface, Transcription Tool Interface) need to be created to interact with these external components (similar to the Facade design pattern). In general, all classes developed support a minimal number of cohesive contracts and there is no unnecessary coupling in the design.

## 2.2 Subsystem and Component Descriptions

|  |  |
| --- | --- |
| **Component Name:** | Log Entry Component |
| **Component Purpose:** | Display, manage, and maintain information about log entries. |
| **Classes:** | Log Entry, Log Entry Manager, Log Entry Configuration |
| **Contracts:** | 1. Retrieve Log Entry Information  Server: Log Entry  2. Modify Log Entry Information  Server: Log Entry  3. Display Log Entries  Server: Log Entry Configuration  4. Retrieve Log Entries  Server: Log Entry Manager  5. Manage Log Entry:  Server Log Entry Manager |
| **Section Discussed:** | Section 3 |

|  |  |
| --- | --- |
| **Component Name:** | Icon Component |
| **Component Purpose:** | Display, manage, and maintain information about icons |
| **Classes:** | Icon, Icon Manager, Icon Configuration |
| **Contracts:** | 27. Retrieve Icon Information  Server: Icon  28. Retrieve Icons  Server: Icon Manager  29. Manage Icon  Server: Icon Manager  30. Display Icon Manager  Server: Icon Configuration |
| **Section Discussed:** | Section 8 |

|  |  |
| --- | --- |
| **Component Name:** | Vector Component |
| **Component Purpose:** | Manages and maintains vector information, graph information, node information and node relationship information. |
| **Classes:** | Vector, Node, Relationship, Graph, Vector Configuration, Graph Configuration |
| **Contracts:** | 31. Retrieve Vector Information  Server: Vector  32. Manage Node  Server: Vector  33. Manage Relationship  Server: Vector  34. Retrieve Graph Information  Server: Graph  35. Manage Graph  Server: Graph  36. Display Vector  Server: Vector Table Configuration  37. Display Graph  Server: Graph Configuration  38. Retrieve Node Information  Server: Node  39. Modify Node Information  Server: Node  40. Retrieve Relationship Information  Server: Relationship  41. Modify Relationship Information  Server: Relationship |
| **Section Discussed:** | Section 9 |

|  |  |
| --- | --- |
| **Subsystem Name:** | Log File Component |
| **Subsystem Purpose:** | Display, manage, and maintain information about log files and enforcement action reports |
| **Classes:** | Log File (Audio Log File, Video Log File, Image Log File, PDF Log File), Enforcement Action Report, Enforcement Action Report Manager, Log File Manager, Log File Configuration, Enforcement Action Report Configuration |
| **Contracts:** | 18. Ingest Log File  Server: Log File  19. Retrieve Log File Information  Server: Log File  20. Retrieve Enforcement Action Report Information  Server: Enforcement Action Report  21. Change Log File Status  Server: Enforcement Action Report  22. Retrieve Enforcement Action Reports  Server: Enforcement Action Report Manager  23. Manage Enforcement Action Report  Server: Enforcement Action Report Manager  24. Retrieve Log Files  Server: Log File Manager  25. Manage Log File  Server: Log File Manager  26. Display Log Files/Enforcement Action Reports  Server: Log File Configuration |
| **Section Discussed:** | Section 7 |

|  |  |
| --- | --- |
| **Component Name:** | Event Configuration Component |
| **Component Purpose:** | Allows the user to view and modify details about the event. |
| **Classes:** | Event Configuration, Event Configuration Window |
| **Contracts:** | 6. Retrieve Event Configuration Information  Server: Event Configuration  7. Modify Event Configuration Information  Server: Event Configuration  8. Initiate Data Ingestion  Server: Event Configuration  9. Display Event Configuration Information  Server: Event Configuration Window |
| **Section Discussed:** | Section 4 |

|  |  |
| --- | --- |
| **Component Name:** | Vector Db Component |
| **Component Purpose:** | Stores vectors that have been defined by the user so they can access, modify, or delete a specified vector. |
| **Classes:** | Vector Db, Vector Db Configuration |
| **Contracts:** | 10. Retrieve Vectors  Server: VectorDb  11. Manage Vector  Server: VectorDb  12. Handle Vector Db Versioning  Server: Vector Db  13. Display Vector Db  Server: VectorDb Configuration |
| **Section Discussed:** | Section 5 |

|  |  |
| --- | --- |
| **Component Name:** | Client/Server Component |
| **Component Purpose:** | Store system information and handle communication between system users |
| **Classes:** | Client Handler, Server Handler |
| **Contracts:** | 14. Send Data  Server: Client Handler  15. Send Get Data Request  Server: Client Handler  16. Handle Received Data  Server: Server Handler  17. Handle Get Data Request  Server: Server Handler |
| **Section Discussed:** | Section 6 |

|  |  |
| --- | --- |
| **Component Name:** | Splunk Interface |
| **Component Purpose:** | Coordinate with Splunk to ingest textual log entries |
| **Classes:** | Splunk Interface |
| **Contracts:** | 42. Retrieve Log Entries From Splunk  Server: Splunk Interface |
| **Section Discussed:** | Section 10 |

|  |  |
| --- | --- |
| **Component Name:** | OCR Tool Interface |
| **Component Purpose:** | Coordinate with the OCR tool to ingest textual log entries |
| **Classes:** | OCR Tool Interface |
| **Contracts:** | 43. Retrieve Log Entries From the OCR Tool  Server: OCR Tool Interface |
| **Section Discussed:** | Section 11 |

|  |  |
| --- | --- |
| **Component Name:** | Transcription Tool Interface |
| **Component Purpose:** | Coordinate with the Transcription tool to ingest textual log entries |
| **Classes:** | Transcription Tool Interface |
| **Contracts:** | 44. Retrieve Log Entries From the Transcription Tool  Server: Transcription Tool Interface |
| **Section Discussed:** | Section 12 |

## 2.3 Dependencies

The components are, for the most part, not very dependent on each other to accomplish their individual responsibilities and in our opinion, there is no unnecessary coupling in our system design. Because of this modularity, the system components should be relatively easy to develop, and even more importantly, they should be relatively easy to integrate. The way that some of the components are dependent on each other may make testing difficult in the future, but we should be able to overcome most testing difficulties using stubs or other workarounds. In general, we do not expect any major issues to occur based on component dependencies.

# 3. Detailed Description of Log Entry Component

**Component Name:** Log Entry Component

**Component Purpose:** Display, manage, and maintain information about log entries

**Classes:** Log Entry, Log Entry Manager, Log Entry Configuration

## 3.1 Log Entry

|  |  |
| --- | --- |
| **Class Name**: Log Entry | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities** | |
| **Contract:** 1. Retrieve Log Entry Information | |
| **Responsibilities** | **Collaborations** |
| 1. Knows the log entry number  2. Knows the log entry timestamp  3. Knows the log entry contents  4. Knows the host IP address  5. Knows the source log file  6. Knows the source type |  |
| **Contract:** 2. Modify Log Entry Information | |
| **Responsibilities** | **Collaborations** |
| 7. Modify the log entry contents |  |

### 3.1.1 Retrieve Log Entry Information

**Contract Name:** Retrieve Log Entry Information

**Contract ID:** 1

**Contract Description:** Retrieves and returns log entry information (log entry number, log entry timestamp, log entry contents, host IP address, source log file, and source type)

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the log entry number |
| **Method Signature:** def getLogEntryNumber() |
| **Pre-Conditions:** Log entry has been initialized |
| **Post-Condition:** The log entry number of the log entry is returned to the client. No log entry information is changed. |
| **Purpose:** Retrieve and return the log entry number of a log entry. |

|  |
| --- |
| **Responsibility:** Knows the log entry timestamp |
| **Method Signature:** def getLogEntryTimestamp() |
| **Pre-Conditions:** Log entry has been initialized |
| **Post-Condition:** The log entry timestamp of the log entry is returned to the client. No log entry information is changed. |
| **Purpose:** Retrieve and return the log entry timestamp of a log entry. |

|  |
| --- |
| **Responsibility:** Knows the log entry contents |
| **Method Signature:** def getLogEntryContents() |
| **Pre-Conditions:** Log entry has been initialized |
| **Post-Condition:** The log entry contents of the log entry is returned to the client. No log entry information is changed. |
| **Purpose:** Retrieve and return the log entry contents of a log entry. |

|  |
| --- |
| **Responsibility:** Knows the log entry host IP address |
| **Method Signature:** def getLogEntryHostIPAddress() |
| **Pre-Conditions:** Log entry has been initialized |
| **Post-Condition:** The host IP address of the log entry is returned to the client. No log entry information is changed. |
| **Purpose:** Retrieve and return the host IP address of a log entry. |

|  |
| --- |
| **Responsibility:** Knows the log entry source log file |
| **Method Signature:** def getLogEntryHostIPAddress() |
| **Pre-Conditions:** Log entry has been initialized |
| **Post-Condition:** The host IP address of the log entry is returned to the client. No log entry information is changed. |
| **Purpose:** Retrieve and return the host IP address of a log entry. |

### 3.1.2 Modify Log Entry Information

**Contract Name:** Modify Log Entry Information

**Contract ID:** 2

**Contract Description:** Modify log entry contents based on provided input.

**Protocol:**

|  |
| --- |
| **Responsibility:** Modify Log Entry Contents |
| **Method Signature:** def setLogEntryContents(newContent) |
| **Pre-Conditions:** Log entry has been initialized. The parameter newContent is a String. |
| **Post-Condition:** The log entry’s content information is changed to the provided input. |
| **Purpose:** Modify log entry contents based on provided input. |

## 3.2 Log Entry Configuration

|  |
| --- |
| **Class Name**: Log Entry Configuration |
| **Superclass**: N/A |
| **Subclasses**: N/A |
| **Private Responsibilities**  1. Modify Log Entry Manager Information |

|  |  |
| --- | --- |
| **Contract:** 3. Display Log Entries | |
| **Responsibilities** | **Collaborations** |
| 2. Display Log Entry Manager Information | Log Entry Manager (4) |

### 3.2.1 Display Log Entries

**Contract Name:** Display Log Entries

**Contract ID:** 3

**Contract Description:** Display log entries in a user interface.

**Protocol:**

|  |
| --- |
| **Responsibility:** Display Log Entries |
| **Method Signature:** def displayLogEntries(logEntries) |
| **Pre-Conditions:** Log Entry Configuration has been initialized. The parameter logEntries is non-null. |
| **Post-Condition:** The provided log entries are displayed on a user interface. |
| **Purpose:** Display log entries in a user interface. |

## 3.3 Log Entry Manager

|  |  |
| --- | --- |
| **Class Name**: Log Entry Manager | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities**   1. Store log entries in a system data store 2. Retrieve log entries in a system data store 3. Display log entries | |
| **Contract:** 4. Retrieve Log Entries | |
| **Responsibilities** | **Collaborations** |
| 4. Knows a list of searched log entries.  5. Search system data store for log entries  6. Sort log entries | Log Entry (1)  Client Handler (6) |
| **Contract:** 5. Manage Log Entry | |
| **Responsibilities** | **Collaborations** |
| 7. Associate a log entry to a vector.  8. Edit log entry  9. Create log entry | Vector (8)  Log Entry (7) |

### 3.3.1 Retrieve Log Entries

**Contract Name:** Retrieve Log Entries

**Contract ID:** 4

**Contract Description:** Retrieves and returns log entries to display to the user

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows a list of searched log entries. |
| **Method Signature:** def getSearchedLogEntries() |
| **Pre-Conditions:** The log entry manager has been initialized. |
| **Post-Condition:** The list of searched log entries is returned to the user. The log entry manager data remains unchanged. |
| **Purpose:** Retrieve and return the list of searched log entries. |

|  |
| --- |
| **Responsibility:** Search system data store for log entries. |
| **Method Signature:** def searchLogEntries(commandSearch, creatorBlueTeam, creatorWhiteTeam, creatorRedTeam, eventTypeBlueTeam, eventTypeWhiteTeam, eventTypeRedTeam, startTime, endTime, locationSearch) |
| **Pre-Conditions:** The log entry manager must be initialized. The client handler must be connected to the server handler over the network. |
| **Post-Condition:** The log entry manager’s list of searched log entries is updated. The stored log entries remain unchanged. |
| **Purpose:** Search the system datastore for log entries based on provided search criteria. |

|  |
| --- |
| **Responsibility:** Sort log entries |
| **Method Signature:** def sortLogEntries(field, reverse) |
| **Pre-Conditions:** The log entry manager must be initialized. The log entry manager must have a list of log entries that has length greater than 0. The field parameter is a string that reflects a valid log entry field. The reverse parameter is Boolean. |
| **Post-Condition:** The log entry manager’s list of searched log entries is updated to reflect the sort criteria entered in the parameters. |
| **Purpose:** Sort the log entries in the log entry manager based on the sorting criteria. |

### 3.3.2 Manage Log Entry

**Contract Name:** Manage Log Entry

**Contract ID:** 5

**Contract Description:** Edit or create a log entry as well as associate a log entry to a vector.

**Protocol:**

|  |
| --- |
| **Responsibility:** Edit an existing log entry. |
| **Method Signature:** def editLogEntry(newLogEntry) |
| **Pre-Conditions:** The log entry manager must be initialized. The logEntry parameter must be initialized and have a valid identifier. |
| **Post-Condition:** The log entry with the same identifier as newLogEntry is updated to have the same fields as newLogEntry. |
| **Purpose:** Edits a log entry based on provided parameter. |

|  |
| --- |
| **Responsibility:** Create log entry. |
| **Method Signature:** def createLogEntry(content, timestamp, source, eventType, creator) |
| **Pre-Conditions:** The log entry manager must be initialized. The provided parameters are all non-null. |
| **Post-Condition:** A log entry is created with a new unique identifier and the same fields as provided by the parameters. The log entry is sent to the server and stored. |
| **Purpose:** Creates a new log entry based on provided input. |

|  |
| --- |
| **Responsibility:** Associate a log entry to a vector. |
| **Method Signature:** def associateLogEntryToVector(logEntry, vectorName) |
| **Pre-Conditions:** The log entry must be initialized. The vectorName must be valid (i.e. there must be a vector in the vector manager with the provided name). |
| **Post-Condition:** The vector’s list of log entries is augmented with the provided log entry. The log entry manager’s associations are updated to reflect the change. A significant event is created in the vector for the log entry. |
| **Purpose:** Associate a log entry to a vector based on vector name. |

# 4. Detailed Description of Event Configuration Component

**Component Name:** Event Configuration Component

**Component Purpose:** Display, manage, and maintain information about event configuration

**Classes:** Event Configuration, Event Configuration Window

## 4.1 Event Configuration

|  |  |
| --- | --- |
| **Class Name**: Event Configuration | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities**  1. Store all event configuration information in a system datastore  2. Retrieve all event configuration from a system datastore  3. Validate directory structure  4. Display event configuration information | |
| **Contract:** 6. Retrieve Event Configuration Information | |
| **Responsibilities** | **Collaborations** |
| 5. Knows the name of the event  6. Knows the start timestamp of an event  7. Knows the end timestamp of an event  8. Knows the path to the root directory of an event  9. Knows the name of the red team folder  10. Knows the name of the blue team folder  11. Knows the name of the white team folder  12. Knows the IP address of the lead |  |
| **Contract:** 7. Modify Event Configuration Information | |
| **Responsibilities** | **Collaborations** |
| 13. Modify the name of the event  14. Modify the start timestamp of an event  15. Modify the end timestamp of an event  16. Modify the IP address of the lead  17. Modify event root path |  |
| **Contract:** 8. Initiate Data Ingestion | |
| **Responsibilities** | **Collaborations** |
| 18. Initiate Data Ingestion | Log File Manager (6) |

### 4.1.1 Retrieve Event Configuration Information

**Contract Name:** Retrieve Event Configuration Information

**Contract ID:** 6

**Contract Description:** Retrieves and returns event configuration to display to the user.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the name of the event |
| **Method Signature:** def getEventName() |
| **Pre-Conditions:** Event configuration has been initialized. Event name is not null. |
| **Post-Condition:** The name of the event is returned to the user. |
| **Purpose:** Returns the name of the event to the user |

### 

|  |
| --- |
| **Responsibility:** Knows the start timestamp of an event |
| **Method Signature:** def getEventStartTime() |
| **Pre-Conditions:** Event configuration has been initialized. Event start time is not null |
| **Post-Condition:** The start timestamp is returned to the user. |
| **Purpose:** Returns the start time of the event to the user |

### 

|  |
| --- |
| **Responsibility:** Knows the end timestamp of an event |
| **Method Signature:** def getEventEndTime() |
| **Pre-Conditions:** Event configuration has been initialized. Event end time is not null |
| **Post-Condition:** The event end time is returned to the user. |
| **Purpose:** Returns the end time of the event to the user |

### 

|  |
| --- |
| **Responsibility:** Knows the path to the root directory of an event |
| **Method Signature:** def getRootPath() |
| **Pre-Conditions:** Event configuration has been initialized. Root path is not null |
| **Post-Condition:** The root path is returned to the user. |
| **Purpose:** Returns the root path to the user |

### 

|  |
| --- |
| **Responsibility:** Knows the name of the red team folder |
| **Method Signature:** def getRedTeamDirectory() |
| **Pre-Conditions:** Event configuration has been initialized. The root path has been initialized |
| **Post-Condition:** The red team directory name is returned to the user. |
| **Purpose:** Returns the path for the red team’s log files |

### 

|  |
| --- |
| **Responsibility:** Knows the name of the blue team folder |
| **Method Signature:** def getBlueTeamDirectory() |
| **Pre-Conditions:** Event configuration has been initialized. The root path has been initialized |
| **Post-Condition:** The blue team directory name is returned to the user. |
| **Purpose:** Returns the path for the blue team’s log files |

### 

|  |
| --- |
| **Responsibility:** Knows the name of the white team folder |
| **Method Signature:** def getWhiteTeamDirectory() |
| **Pre-Conditions:** Event configuration has been initialized. The root path has been initialized |
| **Post-Condition:** The white team directory name is returned to the user. |
| **Purpose:** Returns the path for the white team’s log files |

### 

|  |
| --- |
| **Responsibility:** Knows the IP address of the lead |
| **Method Signature:** def getLeadIP() |
| **Pre-Conditions:** Event configuration has been initialized. Lead machine has been set |
| **Post-Condition:** The IP address of the lead is returned to the user. |
| **Purpose:** Returns the IP address of the lead machine |

### 4.1.2 Modify Event Configuration Information

**Contract Name:** Modify Event Configuration Information

**Contract ID:** 7

**Contract Description:** Modify event configuration based on provided input.

**Protocol:**

|  |
| --- |
| **Responsibility:** Modify the name of the event |
| **Method Signature:** def setEventName(newEventName) |
| **Pre-Conditions:** Event configuration has been initialized. Event name is not null. The parameter newEventName is a string. |
| **Post-Condition:** The event name of the event configuration is changed to the provided parameter. |
| **Purpose:** Modifies the name of the event. |

|  |
| --- |
| **Responsibility:** Modify the start timestamp of an event |
| **Method Signature:** def setEventStartTime(newStartTime) |
| **Pre-Conditions:** Event configuration has been initialized. Event start time is not null. The parameter newStartTime is a string that is in the valid datetime format. |
| **Post-Condition:** The start time of the event is changed to the provided parameter. |
| **Purpose:** Modifies the start time of the event. |

|  |
| --- |
| **Responsibility:** Modify the end timestamp of an event |
| **Method Signature:** def setEventEndTime(newEndTime) |
| **Pre-Conditions:** Event configuration has been initialized. Event end time is not null. The parameter newEndTime is a string that is in the valid datetime format. |
| **Post-Condition:** The end time of the event is changed to the provided parameter. |
| **Purpose:** Modifies the end time of the event. |

|  |
| --- |
| **Responsibility:** Modify the IP address of the lead |
| **Method Signature:** def setLeadIP(newIP) |
| **Pre-Conditions:** Event configuration has been initialized. The parameter lead IP is a valid IP address. |
| **Post-Condition:** The lead IP address of the event is changed to the provided parameter if the IP exists on the network. If the IP address does not exists on the network, the lead IP address of the event does not change. |
| **Purpose:** Modifies the IP address of the lead machine |

|  |
| --- |
| **Responsibility:** Modify event root path |
| **Method Signature:** def setRootPath(newRootPath) |
| **Pre-Conditions:** Event configuration has been initialized. The parameter root path is a directory path. |
| **Post-Condition:** The lead IP address of the event is changed to the provided parameter if the IP exists on the network. If the IP address does not exists on the network, the lead IP address of the event does not change. |
| **Purpose:** Modifies the IP address of the lead machine |

### 4.1.3 Initiate Data Ingestion

**Contract Name:** Initiate Data Ingestion

**Contract ID:** 8

**Contract Description:** Initiate data ingestion for all log files.

**Protocol:**

|  |
| --- |
| **Responsibility:** Modify the name of the event |
| **Method Signature:** def initiateDataIngestion() |
| **Pre-Conditions:** Event configuration has been initialized. Log File Manager has been initialized. Enforcement Action Report Manager has been initialized. |
| **Post-Condition:** The log file ingestion process begins asynchronously. |
| **Purpose:** Initiate data ingestion for log files. |

## 4.2 Event Configuration Window

|  |
| --- |
| **Class Name**: Event Configuration Window |
| **Superclass**: N/A |
| **Subclasses**: N/A |
| **Private Responsibilities**  1. Modify event configuration information  2. Request initiation of data ingestion |

|  |  |
| --- | --- |
| **Contract:** 9. Display Event Configuration | |
| **Responsibilities** | **Collaborations** |
| 1. Display Event Configuration information | Event Configuration (6) |

### 4.2.1 Display Event Configuration

**Contract Name:** Display Event Configuration

**Contract ID:** 9

**Contract Description:** Display event configuration in a user interface.

**Protocol:**

|  |
| --- |
| **Responsibility:** Display Event Configuration Information |
| **Method Signature:** def displayEventConfiguration(eventConfiguration) |
| **Pre-Conditions:** Event Configuration Window has been initialized. The parameter event configuration is non-null. |
| **Post-Condition:** The provided event configuration is displayed on a user interface. |
| **Purpose:** Display event configuration in a user interface. |

# 5. Detailed Description of Vector Db Component

**Component Name:** Vector Db Component

**Component Purpose:** Display, manage, and maintain information about a vector db

**Classes:** VectorDb (Lead VectorDb, User VectorDb), Vector Db Configuration

## 5.1 Vector Db

|  |
| --- |
| **Class Name**: Vector Database |
| **Superclass**: N/A |
| **Subclasses**: Lead Vector Database, User Vector Database |
| **Private Responsibilities**  1. Store vectors in a system or local datastore  2. Retrieve vectors from a system or local datastore  3. Display vector Db |

|  |  |
| --- | --- |
| **Contract:** 10. Retrieve Vectors | |
| **Responsibilities** | **Collaborations** |
| 4. Knows a list of vectors  5. Knows a list of pushed vectors  6. Knows a list of pulled vectors | Vector (4)  Client Handler (6)  Client Handler (6) |
| **Contract:** 11. Handle VectorDb Versioning | |
| **Responsibilities** | **Collaborations** |
| 5. Push vector db to lead  6. Pull vector db from lead  7. Approve vector  8. Reject vector | Client Handler (7)  Client Handler (7) |
| **Contract:** 12. Manage Vector | |
| **Responsibilities** | **Collaborations** |
| 9. Create vector  10. Delete vector |  |

### 5.1.1 Retrieve Vectors

**Contract Name:** Retrieve Vectors

**Contract ID:** 10

**Contract Description:** Retrieves and returns vectors to display to users.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows a list of vectors |
| **Method Signature:** def getVectors() |
| **Pre-Conditions:** The Vector Db has been initialized. |
| **Post-Condition:** The list of vectors is returned to the user. |
| **Purpose:** Retrieve and return vectors to the requestor. |

### 5.1.2 Handle Vector Db Versioning

**Contract Name:** Handle Vector Db Versioning

**Contract ID:** 11

**Contract Description:** Handle vector Db versioning between user and lead machines.

**Protocol:**

|  |
| --- |
| **Responsibility:** Push Vector Db to lead |
| **Method Signature:** def pushVectorDb() |
| **Pre-Conditions:** The Vector Db has been initialized. The calling machine is not lead. |
| **Post-Condition:** The current vector Db is pushed to the lead machine for approval or rejection. A copy of the pushed vector db is stored locally. No changes to the vector db happen. |
| **Purpose:** Push current vector Db to lead machine. |

|  |
| --- |
| **Responsibility:** Pull Vector Db from lead |
| **Method Signature:** def pullVectorDb() |
| **Pre-Conditions:** The Vector Db has been initialized. The calling machine is not lead. |
| **Post-Condition:** The current vector Db is updated to the pulled vector db. A copy of the pulled vector db is stored locally. |
| **Purpose:** Pull master vector Db from lead machine. |

|  |
| --- |
| **Responsibility:** Approve vector |
| **Method Signature:** def approveVector(vector) |
| **Pre-Conditions:** The Vector Db has been initialized. The calling machine is lead. The vector parameter is non-null. |
| **Post-Condition:** The current vector Db is updated to include the approved vector. The vector is removed from the list of pushed vectors |
| **Purpose:** Approve vector from user machine. |

|  |
| --- |
| **Responsibility:** Reject vector |
| **Method Signature:** def rejectVector(vector) |
| **Pre-Conditions:** The Vector Db has been initialized. The calling machine is lead. The vector parameter is non-null. |
| **Post-Condition:** The vector is removed from the list of pushed vectors. |
| **Purpose:** Reject vector from user machine. |

### 5.1.3 Manage Vector

**Contract Name:** Manage Vector

**Contract ID:** 12

**Contract Description:** Creates, deletes, or modifies a vector based on provided input. Approval and rejection of sent vectors is also included here.

**Protocol:**

|  |
| --- |
| **Responsibility:** Create new vector. |
| **Method Signature:** def createVector(vectorName, vectorDescription) |
| **Pre-Conditions:** The Vector Db has been initialized. vectorName and vectorDescription are non-null strings. |
| **Post-Condition:** A new vector is created with the same name as the provided parameter if no other vector exists with that name and stored in local storage. If no other vector exists with the same name as vector name, true is returned to the caller, else false. |
| **Purpose:** Create a new vector with the provided vector name and vector description. |

|  |
| --- |
| **Responsibility:** Delete vector. |
| **Method Signature:** def deleteVector(vectorName) |
| **Pre-Conditions:** The Vector Db has been initialized. vectorName is a non-null string and a valid identifier, |
| **Post-Condition:** The vector with a vector name that matches the provided parameter is removed form the vector Db and removed from system storage. No other vectors are affected. |
| **Purpose:** Deletes a vector with the provided vector name. |

## 5.2. Vector Db Configuration

|  |
| --- |
| **Class Name**: Vector Db Configuration |
| **Superclass**: N/A |
| **Subclasses**: N/A |
| **Private Responsibilities**  1. Request manage vector operations  2. Request vector Db versioning operations |

|  |  |
| --- | --- |
| **Contract:** 13. Display VectorDb | |
| **Responsibilities** | **Collaborations** |
| 3. Display vectorDb vectors  4. Display pulled vectors  5. Display pushed vectors | Vector Db (4)  Vector Db (5)  Vector Db (6) |

### 5.2.1 Display Vector Db Configuration

**Contract Name:** Display Vector Db Configuration

**Contract ID:** 13

**Contract Description:** Display vector Db in a user interface.

**Protocol:**

|  |
| --- |
| **Responsibility:** Display vector Db vectors |
| **Method Signature:** def displayVectorDbVectors(vectorDb) |
| **Pre-Conditions:** Vector Db Configuration has been initialized. vectorDb parameter is non-null. |
| **Post-Condition:** The provided vectorDb’s vectors are displayed in a user interface. |
| **Purpose:** Display vectorDb vectors in a user interface. |

|  |
| --- |
| **Responsibility:** Display pulled vectors |
| **Method Signature:** def displayPulledVectors (vectorDb) |
| **Pre-Conditions:** Vector Db Configuration has been initialized. vectorDb parameter is non-null. |
| **Post-Condition:** The pulled vectors of the vectorDb are displayed in a user interface. |
| **Purpose:** Display pulled vectors in a user interface. |

|  |
| --- |
| **Responsibility:** Display pushed vectors |
| **Method Signature:** def displayPushedVectors (vectorDb) |
| **Pre-Conditions:** Vector Db Configuration has been initialized. vectorDb parameter is non-null. |
| **Post-Condition:** The pushed vectors of the vectorDb are displayed in a user interface. |
| **Purpose:** Display pushed vectors in a user interface. |

# 6. Detailed Description of Client/Server Component

**Component Name:** Client/Server Component

**Component Purpose:** Store system information and handle communication between system users

**Classes:** Client Handler/Server Handler

## 6.1 Client Handler

|  |  |
| --- | --- |
| **Class Name**: Client Handler | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities**  1. Knows the IP address of the lead machine  2. Knows the IP address of the server  3. Knows the IP address of a user machine  4. Create a local network connection from the current machine to the server  5. Receive data from the server | |
| **Contract:** 14. Send Data Request | |
| **Responsibilities** | **Collaborations** |
| 6. Send data from client to server |  |
| **Contract:** 15. Send Get Data Request | |
| **Responsibilities** | **Collaborations** |
| 7. Send get data request from client to server |  |

### 6.1.1. Send Data Request

**Contract Name:** Send Data Request

**Contract ID:** 14

**Contract Description:** Sends client data to the server.

**Protocol:**

|  |
| --- |
| **Responsibility:** Process store data request |
| **Method Signature:** def sendDatat(protocol, serializedData) |
| **Pre-Conditions:** The specified protocol is valid. The serialized data is valid. The client handler must be connected to the server handler over the network. |
| **Post-Condition:** A status value is returned that reflects the result of the operation. |
| **Purpose:** Send data to the server by providing client handler with a valid protocol and valid serialized data. |

### 6.1.2. Send Get Data Request

**Contract Name:** Send Get Data Request

**Contract ID:** 15

**Contract Description:** Sends a get data request for client data to the server.

**Protocol:**

|  |
| --- |
| **Responsibility:** Process get data request |
| **Method Signature:** def getDataRequest(protocol) |
| **Pre-Conditions:** The specified protocol is valid. The client handler must be connected to the server handler over the network. |
| **Post-Condition:** A status value is returned that reflects the result of the operation. If the request was successful, the desired data is also returned in a serialized format. |
| **Purpose:** Send a get data request to the server by providing client handler with a valid protocol. |

## 6.2. Server Handler

|  |  |
| --- | --- |
| **Class Name**: Server Handler | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities**  1. Knows the IP address of the server  2. Knows a list of all current server connections from client machines  3. Create a server port to accept connections  4. Send data to connected client machines  5. Receive data from connected client machines  6. Store received data to a system datastore  7. Retrieve data from a system datastore | |
| **Contract:** 16. Handle Received Data | |
| **Responsibilities** | **Collaborations** |
| 8. Handle data sent form a connected client machine | Client Handler (6) |
| **Contract:** 17. Handle Get Data Request | |
| **Responsibilities** | **Collaborations** |
| 9. Retrieve data from a system datastore | Client Handler (7) |

### 6.2.1. Handle Received Data

**Contract Name:** Handle Received Data

**Contract ID:** 16

**Contract Description:** Handles data received from a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Handle data sent form a connected client machine |
| **Method Signature:** def handleReceivedData(protocol, serializedData) |
| **Pre-Conditions:** The specified protocol is valid. The serialized data is in a valid format. The client handler must be connected to the server handler over the network. |
| **Post-Condition:** A status value is returned that reflects the result of the operation. The serialized data is stored in the system datastore, replacing any previous versions of the data. |
| **Purpose:** Handles data sent from a client machine that contains a valid protocol. |

### 6.2.2. Handle Get Data Request

**Contract Name:** Handle Get Data Request

**Contract ID:** 17

**Contract Description:** Handles a get data request from a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Handle Get Data Request |
| **Method Signature:** def handleGetDataRequest(protocol) |
| **Pre-Conditions:** The specified protocol is valid. The client handler must be connected to the server handler over the network. |
| **Post-Condition:** A status value is returned that reflects the result of the operation. If the data exists, the serialized data is returned to the client. |
| **Purpose:** Handles a get data request from a client machine that contains a valid protocol. |

# 7. Detailed Description of Log File Component

**Component Name:** Log File Component

**Component Purpose:** Display, manage, and maintain information about log files and enforcement action reports

**Classes:** Log File (Audio Log File, Video Log File, Image Log File, PDF Log File), Enforcement Action Report, Enforcement Action Report Manager, Log File Manager, Log File Configuration, Enforcement Action Report Configuration

## 7.1 Log File

|  |  |
| --- | --- |
| **Class Name**: Log File | |
| **Superclass**: N/A | |
| **Subclasses**: Audio Log File, Video Log File, Image Log File, PDF Log File | |
| **Private Responsibilities** | |
| **Contract:** 18. Ingest Log File | |
| **Responsibilities** | **Collaborations** |
| 1. Cleanse Log File  2. Validate Log File  3. Ingest Log File | OCR Tool (1) for Image and PDF Log File, Transcription Tool (1) for Video and Audio Log File,  Splunk Interface (1) for generic  Enforcement Action Report Manager (5)  Log Entry Manager (9) |
| **Contract:** 19. Retrieve Log File Information | |
| **Responsibilities** | **Collaborations** |
| 4. Knows the file name of the log file  5. Knows the timestamp of the log file  6. Knows if the file has been cleansed  7. Knows if the file has been validated  8. Knows if the file has been ingested  9. Knows the enforcement action report of the log file | Enforcement Action Report (1) |

### 7.1.1 Ingest Log File

**Contract Name:** Ingest Log File

**Contract ID:** 18

**Contract Description:** Cleanses, validates, and ingests a log file and returns a list of log entries.

**Protocol:**

|  |
| --- |
| **Responsibility:** Cleanse Log File |
| **Method Signature:** def cleanseLogFile(logFile) |
| **Pre-Conditions:** Log file exists |
| **Post-Condition:** The log file becomes cleansed. |
| **Purpose:** Cleanse a log file before validation. |

|  |
| --- |
| **Responsibility:** Validate Log File |
| **Method Signature:** def validateLogFile(logFile) |
| **Pre-Conditions:** Log file exists |
| **Post-Condition:** The log file becomes validated if all the data in the log file is valid. If the data is not valid, an enforcement action report is created that describes the erroneous data. |
| **Purpose:** Validate a log file before ingestion. |

|  |
| --- |
| **Responsibility:** Can ingest a textual log file |
| **Method Signature:** def ingestLogFile(logFile) |
| **Pre-Conditions:** Log file has been validated |
| **Post-Condition:** The log entries from the log files are now in the system |
| **Purpose:** Ingest log file into system. |

### 7.1.2 Retrieve Log File Information

**Contract Name:** Retrieve Log File Information

**Contract ID:** 19

**Contract Description:** Retrieves and returns log file information.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the name of the log file |
| **Method Signature:** def getName() |
| **Pre-Conditions:** The log file has been initialized. |
| **Post-Condition:** The name of the log file is returned to the user. |
| **Purpose:** Returns the name of the log file to the user |

|  |
| --- |
| **Responsibility:** Knows the timestamp of the log file |
| **Method Signature:** def getTimestamp() |
| **Pre-Conditions:** The log file has been initialized. |
| **Post-Condition:** The timestamp of the log file is returned to the user. |
| **Purpose:** Returns the timestamp of the log file to the user |

|  |
| --- |
| **Responsibility:** Knows if the file has been cleansed. |
| **Method Signature:** def isCleansed() |
| **Pre-Conditions:** The log file has been initialized. |
| **Post-Condition:** The cleansing status of the log file as a Boolean is returned to the user. |
| **Purpose:** Returns the cleansing status of the log file as a Boolean to the user. |

|  |
| --- |
| **Responsibility:** Knows if the file has been validated. |
| **Method Signature:** def isValidated() |
| **Pre-Conditions:** The log file has been initialized. |
| **Post-Condition:** The validation status of the log file as a Boolean is returned to the user. If the log file is not able to be validated, an enforcement action report is created. |
| **Purpose:** Returns the validation status of the log file as a Boolean to the user. |

|  |
| --- |
| **Responsibility:** Knows if the file has been ingested. |
| **Method Signature:** def isIngested() |
| **Pre-Conditions:** The log file has been initialized. |
| **Post-Condition:** The ingestion status of the log file as a Boolean is returned to the user. |
| **Purpose:** Returns the ingestion status of the log file as a Boolean to the user. |

|  |
| --- |
| **Responsibility:** Knows the enforcement action report of the log file. |
| **Method Signature:** def getEnforcementActionReport() |
| **Pre-Conditions:** The log file has been initialized. |
| **Post-Condition:** The enforcement action report of the log file is returned to the caller if one exists, none is returned otherwise. |
| **Purpose:** Returns the enforcement action report of the log file as a Boolean to the user. |

## 7.2 Enforcement Action Report

|  |  |
| --- | --- |
| **Class Name**: Enforcement Action Report | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities** | |
| **Contract:** 20. Retrieve Enforcement Action Report Information | |
| **Responsibilities** | **Collaborations** |
| 1. Knows the line number where an error occurred in a log file  2. Knows the error message explaining why a log file fails the validation test  3. Knows the log file the error occurred in |  |
| **Contract:** 21. Change Log File Status | |
| **Responsibilities** | **Collaborations** |
| 4. Accept invalid log file as valid | Log File (3) |

### 7.2.1 Retrieve Enforcement Action Report Information

**Contract Name:** Retrieve Enforcement Action Report Information

**Contract ID:** 20

**Contract Description:** Retrieves and returns enforcement action report information to the caller.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the line number where an error occurred in a log file |
| **Method Signature:** def getLineNumber() |
| **Pre-Conditions:** The enforcement action report has been initialized. |
| **Post-Condition:** The line number known by the enforcement action report is returned to the caller. |
| **Purpose:** Retrieves and returns the line number where an error occurred in a log file to the caller. |

|  |
| --- |
| **Responsibility:** Knows the error message explaining why a log file failed validation |
| **Method Signature:** def getErrorMessage() |
| **Pre-Conditions:** The enforcement action report has been initialized. |
| **Post-Condition:** The error message known by the enforcement action report is returned to the caller. |
| **Purpose:** Retrieves and returns the error message explaining why a log file failed validation to the caller. |

|  |
| --- |
| **Responsibility:** Knows the log file the error occurred in |
| **Method Signature:** def getLogFile() |
| **Pre-Conditions:** The enforcement action report has been initialized. |
| **Post-Condition:** The log file known by the enforcement action report is returned to the caller. |
| **Purpose:** Retrieves and returns the log file the error occurred in to the caller. |

### 7.2.2 Change Log File Status

**Contract Name:** Change Log File Status

**Contract ID:** 21

**Contract Description:** Change log file status to valid, restarting the ingestion process for the log file.

**Protocol:**

|  |
| --- |
| **Responsibility:** Accept invalid log file as valid |
| **Method Signature:** def validateLogFile() |
| **Pre-Conditions:** The enforcement action report has been initialized. The log file known by the enforcement action report is non-null. |
| **Post-Condition:** The log file’s validation status is set to true and the ingestion process for the log file begins. |
| **Purpose:** Change log file status to valid, restarting the ingestion process for the log file. |

## 7.3 Enforcement Action Report Manager

|  |  |
| --- | --- |
| **Class Name**: Enforcement Action Report Manager | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities**  1. Can store enforcement action reports in a local data store  2. Can retrieve enforcement action reports from a local datastore  3. Display Enforcement Action Reports | |
| **Contract:** 22. Retrieve Enforcement Action Reports | |
| **Responsibilities** | **Collaborations** |
| 4. Knows a list of all enforcement action reports created | Enforcement Action Report (1) |
| **Contract:** 23. Manage Enforcement Action Report | |
| **Responsibilities** | **Collaborations** |
| 5. Create a new enforcement action report  6. Validate enforcement action report | Enforcement Action Report (4) |

### 7.3.1 Retrieve Enforcement Action Reports

**Contract Name:** Retrieve Enforcement Action Reports

**Contract ID:** 22

**Contract Description:** Retrieves and returns enforcement action reports so they can be displayed.

**Protocol:**

|  |
| --- |
| **Responsibility:** Retrieve Enforcement Action Report |
| **Method Signature:** def getEnforcementActionReport() |
| **Pre-Conditions:** The enforcement action report manager is initialized. |
| **Post-Condition:** The list of enforcement action reports is retrieved and returned to the caller. |
| **Purpose:** Retrieve and return known enforcement action reports. |

### 7.3.2 Manage Enforcement Action Report

**Contract Name:** Retrieve Enforcement Action Report

**Contract ID:** 23

**Contract Description:** Creates or validates an enforcement action report.

**Protocol:**

|  |
| --- |
| **Responsibility:** Create a new enforcement action report |
| **Method Signature:** def createEnforcementActionReport(lineNumber, errorMessage, logFile) |
| **Pre-Conditions:** The enforcement action report manager is initialized. |
| **Post-Condition:** A new enforcement action report is created with the same fields as the provided parameters and the new enforcement action report is stored in a local datastore. The enforcement action report is returned to the caller. |
| **Purpose:** Create a new enforcement action report based on provided input. |

## 7.4 Log File Manager

|  |  |
| --- | --- |
| **Class Name**: Log File Manager | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities**  1. Can store log files in a local data store  2. Can retrieve log files from a local data store  3. Display log files | |
| **Contract:** 24. Retrieve Log Files | |
| **Responsibilities** | **Collaborations** |
| 4. Knows a list of all log files created | Log File (4) |
| **Contract:** 25. Manage Log File | |
| **Responsibilities** | **Collaborations** |
| 5. Create a new log file  6. Start log file ingestion | Log File (3) |

### 7.4.1 Retrieve Log Files

**Contract Name:** Retrieve Log Files

**Contract ID:** 24

**Contract Description:** Retrieves and returns log files so they can be displayed.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows a list of all log files created |
| **Method Signature:** def getLogFiles() |
| **Pre-Conditions:** The log file manager must be initialized. |
| **Post-Condition:** The list of log files known to the log file manager are retrieved and returned to the caller. |
| **Purpose:** Retrieve and return all log files to the caller. |

### 7.4.2 Manage Log File

**Contract Name:** Manage Log File

**Contract ID:** 25

**Contract Description:** Creates new log files or starts log file ingestion for an existing log file.

**Protocol:**

|  |
| --- |
| **Responsibility:** Create a new log file |
| **Method Signature:** def createLogFile(source) |
| **Pre-Conditions:** The log file manager must be initialized. Source must be a valid full path name. |
| **Post-Condition:** A new log file is created with fields based on its source directory and source name. |
| **Purpose:** Create a new log file based on provided input. |

|  |
| --- |
| **Responsibility:** Start log file ingestion |
| **Method Signature:** def startIngestion(source) |
| **Pre-Conditions:** The log file manager must be initialized. Source must be a valid full path name and a valid log file identifier. |
| **Post-Condition:** The ingestion process for the log file begins. |
| **Purpose:** Starts log file ingestion for log file associated to provided source identifier. |

## 7.5. Log File Configuration

|  |
| --- |
| **Class Name**: Log File Configuration |
| **Superclass**: N/A |
| **Subclasses**: N/A |
| **Private Responsibilities**  1. Request manage log file operations  2. Request manage enforcement action report operations |

|  |  |
| --- | --- |
| **Contract:** 26. Display Log Files/Enforcement Action Reports | |
| **Responsibilities** | **Collaborations** |
| 3. Display Enforcement Action Report Manager Information  4. Display Log File Manager Information | Enforcement Action Report Manager (4)  Log File Manager (4) |

### 7.5.1 Display Log Files/Enforcement Action Reports

**Contract Name:** Display Log Files/Enforcement Action Reports

**Contract ID:** 26

**Contract Description:** Display log files or enforcement action reports in a user interface.

**Protocol:**

|  |
| --- |
| **Responsibility:** Display Enforcement Action Report Manager Information |
| **Method Signature:** def displayEnforcementReportManager(enforcementReportManager) |
| **Pre-Conditions:** Log File Configuration has been initialized. enforcementReportManager parameter is non-null. |
| **Post-Condition:** The provided enforcementReportManager is displayed on a user interface. |
| **Purpose:** Display Enforcement Action Report Manager in a user interface. |

|  |
| --- |
| **Responsibility:** Display Log File Manager Information |
| **Method Signature:** def displayLogFileManager(logFileManager) |
| **Pre-Conditions:** Log File Configuration has been initialized. logFileManager parameter is non-null. |
| **Post-Condition:** The provided logFileManager is displayed on a user interface. |
| **Purpose:** Display Log File Manager in a user interface. |

# 8. Detailed Description of Icon Component

**Component Name:** Icon Component

**Component Purpose:** Display, manage, and maintain information about icons

**Classes:** Icon, Icon Manager, Icon Configuration

## 8.1 Icon

|  |  |
| --- | --- |
| **Class Name**: Icon | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**  1. Generate image from filepath  2. Generate pixel map from filepath | |
| **Contract:** 27. Retrieve Icon Information | |
| **Responsibilities** | **Collaborations** |
| 3. Knows the name of the icon  4. Knows the file path of the icon  5. Knows the image of the icon  6. Knows the pixel map for an icon |  |

### 8.1.1 Retrieve Icon Image

**Contract Name:** Retrieve Icon Image

**Contract ID:** 27

**Contract Description:** Retrieves and returns an icon image for a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the name of the icon |
| **Method Signature:** def getName() |
| **Pre-Conditions:** The icon has been initialized. |
| **Post-Condition:** The icon name is retrieved and returned to the caller. None of the icon attributes are changed. |
| **Purpose:** Retrieve and return the icon name to the caller. |

|  |
| --- |
| **Responsibility:** Knows the fie path of the icon |
| **Method Signature:** def getFilepath() |
| **Pre-Conditions:** The icon has been initialized. |
| **Post-Condition:** The icon filepath is retrieved and returned to the caller. None of the icon attributes are changed. |
| **Purpose:** Retrieve and return the icon filepath to the caller. |

|  |
| --- |
| **Responsibility:** Knows the image of the icon |
| **Method Signature:** def getPixmapByteArray() |
| **Pre-Conditions:** The icon pixmapByteArray has been initialized from the source image file. |
| **Post-Condition:** Streams the image to the GUI of the system. None of the icon attributes are changed. |
| **Purpose:** Used for streaming icon images onto the attack graph that clients will use to represent different types of events. |

|  |
| --- |
| **Responsibility:** Knows the pixel map for an icon |
| **Method Signature:** def getGraphImage() |
| **Pre-Conditions:** The icon image has been initialized from the source image file. |
| **Post-Condition:** Opens the image file as a separate popup window. None of the icon attributes are changed. |
| **Purpose:** Used for when the user wants to view what an image looks like before they start representing nodes with the image. |

## 8.2 Icon Manager

|  |  |
| --- | --- |
| **Class Name**: Icon Manager | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**  1. Store icons in a system datastore  2. Retrieve icons from a system datastore  3. Display icons | |
| **Contract:** 28. Retrieve Icons | |
| **Responsibilities** | **Collaborations** |
| 4. Knows a list of all icons created | Icon (3) |
| **Contract:** 29. Manage Icon | |
| **Responsibilities** | **Collaborations** |
| 5. Create a new icon |  |

### 8.2.1 Retrieve Icons

**Contract Name:** Retrieve Icons

**Contract ID:** 28

**Contract Description:** Retrieves and returns a list of icons for a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows a list of all icons created |
| **Method Signature:** def retrieveIcons(self) |
| **Pre-Conditions:** The icon manager has been initialized. |
| **Post-Condition:** If there exists a pkl file that holds all icons, then retrieve all the icons and put them into an attribute of Icon Manager. None of the icons are changed either in the pkl file or in the list attribute. |
| **Purpose:** Used for when the system is restarting and needs to check if icons have been created during a previous session. After this, the list of icons can be accessed. |

### 8.2.2 Manage Icon

**Contract Name:** Manage Icon

**Contract ID:** 29

**Contract Description:** Creates a new icon based on provided input.

**Protocol:**

|  |
| --- |
| **Responsibility:** Create a new icon |
| **Method Signature:** def createIcon(name, filename) |
| **Pre-Conditions:** The icon manager has been initialized. Name is a unique identifier that has no collisions with other icon names and filename is a valid full filename with an image extension. |
| **Post-Condition:** A new icon is created with fields based on the provided input and stored in system storage. |
| **Purpose:** Create a new icon based on provided input |

## 8.3. Icon Configuration

|  |  |
| --- | --- |
| **Class Name**: Icon Configuration | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**  1. Request icon manager operations | |
| **Contract:** 30. Display Icon Manager | |
| **Responsibilities** | **Collaborations** |
| 2. Display Icon Manager Information | Icon Manager (4) |

### 8.3.1 Display Icon Manager

**Contract Name:** Display Icon Manager

**Contract ID:** 30

**Contract Description:** Display icons in a user interface.

**Protocol:**

|  |
| --- |
| **Responsibility:** Display Icon Manager Information |
| **Method Signature:** def displayIconManager(iconManager) |
| **Pre-Conditions:** Icon Configuration has been initialized. iconManager parameter is non-null. |
| **Post-Condition:** The provided iconManager is displayed on a user interface. |
| **Purpose:** Display Icon Manager in a user interface. |

# 9. Detailed Description of Vector Component

**Component Name:** Vector Component

**Component Purpose:** Display, manage, and maintain information about a vector

**Classes:** Vector, Node, Relationship, Graph, Vector Configuration, Graph Configuration

## 9.1 Vector

|  |  |
| --- | --- |
| **Class Name**: Vector | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**  1. Store vector in a system datastore.  2. Retrieve vector from a system datastore  3. Display vector | |
| **Contract:** 31. Retrieve Vector Information | |
| **Responsibilities** | **Collaborations** |
| 4. Knows the list of nodes that make up the vector, which are the associated log entries  5. Knows the list of relationships between nodes in the vector.  6. Knows the name of the vector  7. Knows the description of the vector | Node (1)  Relationship (1) |
| **Contract:** 32. Manage Node | |
| **Responsibilities** | **Collaborations** |
| 8. Create a new node  9. Modify an existing node  10. Delete an existing node | Node (10) |
| **Contract:** 33. Manage Relationship | |
| **Responsibilities** | **Collaborations** |
| 11. Create a new relationship  12. Modify an existing relationship  13. Delete an existing relationship | Relationship (4) |

## 

### 9.1.1. Retrieve Vector Information

**Contract Name:** Retrieve Vector Information

**Contract ID:** 31

**Contract Description:** Retrieves and returns vector information for a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the user-provided name of the vector. |
| **Method Signature:** def getVectorName() |
| **Pre-Conditions:** Vector has been initialized. |
| **Post-Condition:** The user-provided name of the vector is returned to the client. No vector information is changed in the process. |
| **Purpose:** Retrieve and return the user-provided name of a vector to a client. |

|  |
| --- |
| **Responsibility:** Knows the user-provided description of the vector. |
| **Method Signature:** def getVectorDescription() |
| **Pre-Conditions:** Vector has been initialized. |
| **Post-Condition:** The user-provided description of the vector is returned to the client. No vector information is changed in the process. |
| **Purpose:** Retrieve and return the user-provided description of a vector to a client. |

|  |
| --- |
| **Responsibility:** Knows the list of nodes that make up the vector. |
| **Method Signature:** def getNodes() |
| **Pre-Conditions:** Vector has been initialized. |
| **Post-Condition:** A list of nodes that make up the vector are retrieved and returned to the client. No vector information is changed in the process. |
| **Purpose:** Retrieve and return a list of the vector’s nodes to a client. |

|  |
| --- |
| **Responsibility:** Knows the list of relationships that make up the vector. |
| **Method Signature:** def getRelationships() |
| **Pre-Conditions:** Vector has been initialized. |
| **Post-Condition:** A list of relationships that make up the vector are retrieved and returned to the client. No vector information is changed in the process. |
| **Purpose:** Retrieve and return a list of the vector’s relationships to a client. |

### 9.1.2 Manage Node

**Contract Name:** Mange Nodes

**Contract ID:** 32

**Contract Description:** Creates a new node, deletes an existing node, or modifies an existing node based on provided input.

**Protocol:**

|  |
| --- |
| **Responsibility:** Create a new node. |
| **Method Signature:** def addSignificantEventFromLogEntry(logEntry) |
| **Pre-Conditions:** Log Entry used to create the node has been initialized. The vector has been initialized. |
| **Post-Condition:** A node is created and added to the vector. |
| **Purpose:** Creates a new node from a given log entry and adds it to the vector. |

|  |
| --- |
| **Responsibility:** Modify an existing node. |
| **Method Signature:** def updateSignificantEvent(signficantEvent) |
| **Pre-Conditions:** signficiantEvent passed to the method has been initialized, the significantEvent ID matches to a node that exists in the vector. The vector has been initialized. |
| **Post-Condition:** Node is updated to reflect user-defined modifications. |
| **Purpose:** Modifies a node in the vector using the ID of the significant event it represents. |

|  |
| --- |
| **Responsibility:** Delete an existing node. |
| **Method Signature:** def removeSignificantEventByLogEntry(logEntryId) |
| **Pre-Conditions:** The Log Entry ID matches to a node that exists in a vector. The vector has been initialized. |
| **Post-Condition:** Node is deleted and all of the node’s relationships are deleted as well. |
| **Purpose:** Deletes a node and its relationships from the vector. |

### 9.1.3 Manage Relationship

**Contract Name:** Manage Relationships

**Contract ID:** 33

**Contract Description:** Creates a new relationship, deletes an existing relationship, or modifies an existing relationship based on provided input.

**Protocol:**

|  |
| --- |
| **Responsibility:** Create a new relationship. |
| **Method Signature:** def addNewRelationship(sourceId, destId, description) |
| **Pre-Conditions:** The source node and destination node have been initialized, valid node IDs are provided. The vector has been initialized. |
| **Post-Condition:** A new relationship between two given nodes is created and added to the vector with fields that are identical to the provided parameters. The new relationship is stored to local storage. |
| **Purpose:** Creates a new relationship between two nodes and adds relationship to the vector. |

|  |
| --- |
| **Responsibility:** Modify an existing relationship. |
| **Method Signature:** def updateRelationshipDescription(sourceId, destId, updatedDescription) |
| **Pre-Conditions:** The source node and destination node have been initialized; valid node IDs are provided. sourceId and destId describe a valid relationship in the vector. The vector has been initialized. |
| **Post-Condition:** The relationship’s description defined by the provided sourceId and destId is updated based on provided input. |
| **Purpose:** Update an existing relationship’s description. |

|  |
| --- |
| **Responsibility:** Delete an existing relationship. |
| **Method Signature:** def deleteRelationship(sourceId, destId) |
| **Pre-Conditions:** Valid node IDs are provided. sourceId and destId describe a valid relationship in the vector. The vector has been initialized. |
| **Post-Condition:** The relationship is deleted from the vector and removed from local storage. |
| **Purpose:** Delete an existing relationship in a vector. |

## 9.2. Graph

|  |  |
| --- | --- |
| **Class Name**: Graph | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**  1. Display graph | |
| **Contract:** 34. Retrieve Graph Information | |
| **Responsibilities** | **Collaborations** |
| 2. Knows the vector on which the graph is based | Vector (4) |
| **Contract:** 35. Manage Graph | |
| **Responsibilities** | **Collaborations** |
| 3. Edit the positions of the nodes.  4. Associate nodes together as relationships | Vector (9)  Vector (11) |

### 9.2.1 Retrieve Graph Information

**Contract Name:** Retrieve Graph Information

**Contract ID:** 34

**Contract Description:** Retrieve graph information (vector that a graph is based on) for a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows vector that makes up the graph. |
| **Method Signature:** def getVector() |
| **Pre-Conditions:** Graph has been initialized. The vector the graph is based on has been initialized. |
| **Post-Condition:** The vector that makes up the graph is returned to the client. No graph information is changed in the process. |
| **Purpose:** Retrieve and return the vector that makes up the graph to a client. |

### 9.2.2 Manage Graph

**Contract Name:** Manage Graph

**Contract ID:** 35

**Contract Description:** Edits the position of nodes, associates nodes together as relationships, or exports the nodes and relationships as an image.

**Protocol:**

|  |
| --- |
| **Responsibility:** Edit the positions of nodes. |
| **Method Signature:** def editNodePosition(significantEventId, xPosition, yPosition) |
| **Pre-Conditions:** Graph has been initialized. significantEventId matches the id of one of the nodes in the graph. |
| **Post-Condition:** The position of the node with the same id as significantEventId is changed based on the provided xPosition and yPosition. |
| **Purpose:** Edit the positions of a node that exists in the vector. |

|  |
| --- |
| **Responsibility:** Associate nodes together as relationships. |
| **Method Signature:** def associateNodes(sourceSignificantEvent, destSignificantEvent) |
| **Pre-Conditions:** Graph has been initialized. sourceSignificantEventId matches the id of one of the nodes in the graph and destSignificantEventId matches the id of one of the nodes in the graph |
| **Post-Condition:** A new relationship is created based on the provided sourceSignificantEvent and destSignificantEvent. The relationship is added to the vector on which the graph is based and stored in local storage. |
| **Purpose:** Retrieve and return the list of relationships that make up the graph to a client. |

## 9.3. Vector Table Configuration

|  |  |
| --- | --- |
| **Class Name**: Vector Table Configuration | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**  1. Request vector operations  2. Export vector in a tabular format | |
| **Contract:** 36. Display Vector | |
| **Responsibilities** | **Collaborations** |
| 3. Display Vector Node Information in a tabular format  4. Display Vector Relationship Information in a tabular format | Vector (4)  Vector (5) |

### 9.3.1 Display Vector

**Contract Name:** Display Vector

**Contract ID:** 36

**Contract Description:** Display vector in a user interface.

|  |
| --- |
| **Responsibility:** Display Vector Node Information in a tabular format |
| **Method Signature:** def displayVectorNodes(vector) |
| **Pre-Conditions:** Vector Table Configuration has been initialized. vector parameter is non-null. |
| **Post-Condition:** The provided vector’s nodes are displayed in a user interface in a tabular format. |
| **Purpose:** Display vector’s nodes in a user interface in a tabular format. |

|  |
| --- |
| **Responsibility:** Display Vector Relationship Information in a tabular format |
| **Method Signature:** def displayVectorRelationships(vector) |
| **Pre-Conditions:** Vector Table Configuration has been initialized. vector parameter is non-null. |
| **Post-Condition:** The provided vector’s relationships are displayed in a user interface in a tabular format. |
| **Purpose:** Display vector’s relationships in a user interface in a tabular format. |

## 9.4. Graph Configuration

|  |  |
| --- | --- |
| **Class Name**: Graph Configuration | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**  1. Request Graph operations  2. Export the nodes and relationships as an image. | |
| **Contract:** 37. Display Graph | |
| **Responsibilities** | **Collaborations** |
| 3. Display Graph Information | Graph (2) |

### 9.4.1 Display Graph

**Contract Name:** Display Graph

**Contract ID:** 37

**Contract Description:** Display graph in a user interface.

**Protocol:**

|  |
| --- |
| **Responsibility:** Display Graph Information |
| **Method Signature:** def displayGraph(graph) |
| **Pre-Conditions:** Graph Configuration has been initialized. graph parameter is non-null. |
| **Post-Condition:** The provided graph is displayed on a user interface. |
| **Purpose:** Display graph in a user interface. |

## 9.5. Node

|  |  |
| --- | --- |
| **Class Name**: Node | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:** | |
| **Contract:** 38. Retrieve Node Information | |
| **Responsibilities** | **Collaborations** |
| 1. Knows the node ID  2. Knows the node name  3. Knows the node timestamp  4. Knows the node description  5. Knows the log entry reference  6. Knows the node’s icon  7. Knows the node’s position on the graph  8. Knows a list of attribute visibilities for the node  9. Knows the node’s visibility on a graph | Log Entry (1)  Icon (3) |
| **Contract:** 39. Modify Node Information | |
| **Responsibilities** | **Collaborations** |
| 10. Modify the node name  11. Modify the node description  12. Modify the node’s icon  13. Modify the node’s position on the graph  14. Modify the node’s list of attribute visibilities  15. Modify the node’s visibility on a graph |  |

### 9.5.1 Retrieve Node Information

**Contract Name:** Retrieve Node Information

**Contract ID:** 38

**Contract Description:** Retrieves and returns node information.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the ID of the node. |
| **Method Signature:** def getNodeID() |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The ID of the node is returned to the client. No node information is changed in the process |
| **Purpose:** Retrieve and return the ID of the node to a client. |

|  |
| --- |
| **Responsibility:** Knows the name of the node. |
| **Method Signature:** def getNodeName() |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The name of the node is returned to the client. No node information is changed in the process |
| **Purpose:** Retrieve and return the name of the node to a client. |

|  |
| --- |
| **Responsibility:** Knows the description of the node. |
| **Method Signature:** def getNodeDescription() |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The description of the node is returned to the client. No node information is changed in the process |
| **Purpose:** Retrieve and return the description of the node to a client. |

|  |
| --- |
| **Responsibility:** Knows the timestamp of the node. |
| **Method Signature:** def getNodeTimestamp() |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The timestamp of the node is returned to the client. No node information is changed in the process |
| **Purpose:** Retrieve and return the timestamp of the node to a client. |

|  |
| --- |
| **Responsibility:** Knows the log entry reference of the node. |
| **Method Signature:** def getNodeReference() |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The log entry reference of the node is returned to the client. No node information is changed in the process |
| **Purpose:** Retrieve and return the log entry reference of the node to a client. |

|  |
| --- |
| **Responsibility:** Knows the icon of the node. |
| **Method Signature:** def getNodeIcon() |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The icon of the node is returned to the client. No node information is changed in the process |
| **Purpose:** Retrieve and return the icon of the node to a client. |

|  |
| --- |
| **Responsibility:** Knows the position of the node. |
| **Method Signature:** def getNodePosition() |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The position of the node is returned to the client. No node information is changed in the process |
| **Purpose:** Retrieve and return the position of the node to a client. |

|  |
| --- |
| **Responsibility:** Knows the list of attribute visibilities for the node. |
| **Method Signature:** def getAttributeVisibilities() |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The attribute visibilities of the node is returned to the client. No node information is changed in the process |
| **Purpose:** Retrieve and return the attribute visibilities of the node to a client. |

|  |
| --- |
| **Responsibility:** Knows the node’s visibility. |
| **Method Signature:** def getNodeVisibility() |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The visibility of the node on the graph is returned to the client. No node information is changed in the process |
| **Purpose:** Retrieve and return the visibility of the node on the graph to a client. |

### 9.5.2 Modify Node Information

**Contract Name:** Modify Node Information

**Contract ID:** 39

**Contract Description:** Modifies node information based on provided input.

**Protocol:**

|  |
| --- |
| **Responsibility:** Modify the name of the node. |
| **Method Signature:** def setNodeName(name) |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The name of the node is changed to the provided name. No other node information is changed in the process. |
| **Purpose:** Modifies the name of the node for a client. |

|  |
| --- |
| **Responsibility:** Modify the description of the node. |
| **Method Signature:** def setNodeDescription(description) |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The description of the node is changed to the provided description. No other node information is changed in the process. |
| **Purpose:** Modifies the description of the node for a client. |

|  |
| --- |
| **Responsibility:** Modify the icon of the node. |
| **Method Signature:** def setNodeIcon(newIcon) |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The icon of the node is changed to the provided icon. No other node information is changed in the process. |
| **Purpose:** Modifies the icon of the node for a client. |

|  |
| --- |
| **Responsibility: Modify** the position of the node. |
| **Method Signature:** def setNodePosition(xPosition, yPosition) |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The position of the node is changed to the provided xPosition and yPosition respectively. No other node information is changed in the process. |
| **Purpose:** Modifies the position of the node to a client. |

|  |
| --- |
| **Responsibility:** Modify the list of attribute visibilities for the node. |
| **Method Signature:** def setAttributeVisibilities(fieldname, visibility) |
| **Pre-Conditions:** Node has been initialized. Fieldname is a valid fieldname and visibility is a Boolean. |
| **Post-Condition:** The list of attribute visibilities is changed to reflect the provided input. No other node information is changed in the process. |
| **Purpose:** Modifies the attribute visibilities of the node to a client. |

|  |
| --- |
| **Responsibility:** Modifies the node’s visibility. |
| **Method Signature:** def setNodeVisibility(visibility) |
| **Pre-Conditions:** Node has been initialized. Visibility is a Boolean value. |
| **Post-Condition:** The visibility of the node on the graph is changed based on the provided input. No node information is changed in the process |
| **Purpose:** Modifies the visibility of the node on the graph to a client. |

## 9.6. Relationship

|  |  |
| --- | --- |
| **Class Name**: Relationship | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:** | |
| **Contract:** 40. Retrieve Relationship Information | |
| **Responsibilities** | **Collaborations** |
| 1. Knows the source node ID  2. Knows the destination node ID  3. Knows the relationship description |  |
| **Contract:** 41. Modify Relationship Information | |
| **Responsibilities** | **Collaborations** |
| 4. Modify relationship description |  |

### 9.6.1 Retrieve Relationship Information

**Contract Name:** Retrieve Relationship Information

**Contract ID:** 40

**Contract Description:** Retrieves and returns relationship information.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the source node ID. |
| **Method Signature:** def getSourceNodeID() |
| **Pre-Conditions:** Relationship has been initialized. |
| **Post-Condition:** The source node ID of the relationship returned to the client. No relationship information is changed in the process |
| **Purpose:** Retrieve and return the source node ID of the relationship to a client. |

|  |
| --- |
| **Responsibility:** Knows the destination node ID. |
| **Method Signature:** def getDestinationNodeID() |
| **Pre-Conditions:** Relationship has been initialized. |
| **Post-Condition:** The destination node ID of the relationship returned to the client. No relationship information is changed in the process |
| **Purpose:** Retrieve and return the destination node ID to a client. |

|  |
| --- |
| **Responsibility:** Knows the relationship description. |
| **Method Signature:** def getDescription () |
| **Pre-Conditions:** Relationship has been initialized. |
| **Post-Condition:** The description of the relationship returned to the client. No relationship information is changed in the process |
| **Purpose:** Retrieve and return the relationship description to a client. |

### 9.6.2 Modify Relationship Information

**Contract Name:** Modify Relationship Information

**Contract ID:** 41

**Contract Description:** Modifies relationship information based on provided input.

**Protocol:**

|  |
| --- |
| **Responsibility:** Modify the description of the relationship. |
| **Method Signature:** def setDescription(description) |
| **Pre-Conditions:** Node has been initialized. |
| **Post-Condition:** The description of the relationship is changed to the provided description. No other relationship information is changed in the process. |
| **Purpose:** Modifies the description of the relationship for a client. |

# 10. Detailed Description of Splunk Interface Component

**Component Name:** Splunk Interface Component

**Component Purpose:** Coordinate with Splunk to ingest textual log entries

**Classes:** Splunk Interface

## 10.1. Splunk Interface

|  |  |
| --- | --- |
| **Class Name**: Splunk Interface | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities** | |
| **Contract:** 42. Retrieve Log Entries From Splunk | |
| **Responsibilities** | **Collaborations** |
| 1. Send log files to Splunk for ingestion  2. Retrieve ingested log entries from Splunk |  |

### 10.1.1. Retrieve Log Entries From Splunk

**Contract Name:** Retrieve Log Entries From Splunk

**Contract ID:** 42

**Contract Description:** Sends log files to splunk for ingestion and retrieves the resulting log entries..

**Protocol:**

|  |
| --- |
| **Responsibility:** Send log files to Splunk for ingestion. |
| **Method Signature:** def sendFilesToSplunk(logFiles) |
| **Pre-Conditions:** The splunk interface must be connected to Splunk. The log files must be cleansed and validated. There must be no other requests to Splunk. |
| **Post-Condition:** The sent log files are received by Splunk and ingested asynchronously. |
| **Purpose:** Send cleansed and validated log files to Splunk, which ingests (parses essentially) each individual log file. |

|  |
| --- |
| **Responsibility:** Retrieve ingested log entries from Splunk. |
| **Method Signature:** def retrieveLogEntriesFromSplunk() |
| **Pre-Conditions:** The splunk interface must be connected to Splunk. There must be no other requests to Splunk. |
| **Post-Condition:** The retrieved log entries are returned to the client. No log entry data is modified in the process. |
| **Purpose:** Retrieve all log entries from Splunk ingested from log files and convert them into the log entry format supported by the system. |

# 11. Detailed Description of OCR Tool Interface Component

**Component Name:** OCR Tool Interface Component

**Component Purpose:** Coordinate with the OCR tool to ingest image log entries

**Classes:** OCR Tool Interface

## 11.1. OCR Tool Interface

|  |  |
| --- | --- |
| **Class Name**: OCR Tool Interface | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities** | |
| **Contract:** 43. Retrieve Log Entries From the OCR Tool | |
| **Responsibilities** | **Collaborations** |
| 1. Send log files to the OCR tool for ingestion  2. Retrieve ingested log entries from the OCR tool. |  |

### 11.1. Retrieve Log Entries From the OCR Tool

**Contract Name:** Retrieve Log Entries From the OCR tool

**Contract ID:** 43

**Contract Description:** Sends log files to the OCR tool for ingestion and retrieves the resulting log entries..

**Protocol:**

|  |
| --- |
| **Responsibility:** Send log files to the OCR tool for ingestion. |
| **Method Signature:** def sendFilesToOCRTool(logFiles) |
| **Pre-Conditions:** The ocr tool interface must be connected to the OCR tool. The log files must be cleansed and validated. |
| **Post-Condition:** The sent log files are received by the OCR tool and ingested asynchronously. |
| **Purpose:** Send cleansed and validated log files to the OCR tool, which ingests (parses essentially) each individual log file. |

|  |
| --- |
| **Responsibility:** Retrieve ingested log entries from the OCR tool. |
| **Method Signature:** def retrieveLogEntriesFromOCRTool() |
| **Pre-Conditions:** The ocr tool interface must be connected to the OCR tool. |
| **Post-Condition:** The retrieved log entries are returned to the client. No log entry data is modified in the process. |
| **Purpose:** Retrieve all log entries from the OCR tool ingested from log files and convert them into the log entry format supported by the system. |

# 12. Detailed Description of Transcription Tool Interface Component

**Component Name:** Transcription Tool Interface Component

**Component Purpose:** Coordinate with transcription tool to audio and video log entries.

**Classes:** Transcription Tool Interface

## 12.1. Transcription Tool Interface

|  |  |
| --- | --- |
| **Class Name**: Transcription Tool Interface | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities** | |
| **Contract:** 44. Retrieve Log Entries From the Transcription Tool | |
| **Responsibilities** | **Collaborations** |
| 1. Send log files to the transcription tool for ingestion  2. Retrieve ingested log entries from the transcription tool. |  |

### 12.1. Retrieve Log Entries From the Transcription Tool

**Contract Name:** Retrieve Log Entries From the Transcription tool

**Contract ID:** 44

**Contract Description:** Sends log files to the transcription tool for ingestion and retrieves the resulting log entries.

**Protocol:**

|  |
| --- |
| **Responsibility:** Send log files to the Transcription tool for ingestion. |
| **Method Signature:** def sendFilesToTranscriptionTool(logFiles) |
| **Pre-Conditions:** The transcription tool interface must be connected to the transcription tool. The log files must be cleansed and validated. |
| **Post-Condition:** The sent log files are received by the transcription tool and ingested asynchronously. |
| **Purpose:** Send cleansed and validated log files to the transcription tool, which ingests (parses essentially) each individual log file. |

|  |
| --- |
| **Responsibility:** Retrieve ingested log entries from the transcription tool. |
| **Method Signature:** def retrieveLogEntriesFromTranscriptionTool() |
| **Pre-Conditions:** The transcription tool interface must be connected to the transcription tool. |
| **Post-Condition:** The retrieved log entries are returned to the client. No log entry data is modified in the process. |
| **Purpose:** Retrieve all log entries from the transcription tool ingested from log files and convert them into the log entry format supported by the system. |

# 13. Database Schema

In this section, we describe the database schema we will be using in this project. We use the non-relational database MongoDb to store data in this project. Despite the fact that one of the components is called the Vector Db (Database), it is important to note that we only store log entries in our MongoDb database. We simply use Vector Db as the naming convention because that is what the abstraction is entitled in the SRS. Other forms of data in our system including icons, vectors, log files, and enforcement action reports are serialized to disk and retrieved from disk respectively. They are not stored in any type of database, relational or non-relational. Our database schema is as follows:

**Log Entry Document**

{

"\_id": <id>,

"vectors": ‘[“a”, “b”, “c”]’,

"location": “testLocation”,

"eventType": “testEventType”,

"description": “Lorum ipsum”,

"creator": “testCreator”,

"date": “1/1/2000 12:00 AM”,

"artifact": “testArtifact”

}

To reiterate, our database schema only contains a Log Entry Document because we use other storage mechanisms for system data that is not of type log entry. These storage mechanisms were approved by the client during demos.

**$**