PICK

**Software Design Document**

1.0

3/7/2020

**Document Control**

**Approval**

The Guidance Team and the customer shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 1.0 |
| Indicator of Last Page in Document: | $ |
| Date of Last Review: | 3/9/2020 |
| Date of Next Review: | 3/8/2020 |
| Target Date for Next Update: | 3/8/2020 |

**Distribution List**

This following list of people shall receive a copy of this document every time a new version of this document becomes available: Guidance Team Members, 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 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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). 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**

## 1.3 **Definitions, Acronyms, and Abbreviations**

### 1.3.1 **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. |
| 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. |
| 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. |

### 

### 1.3.2 **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 |

### 

### 1.3.3 **Abbreviations**

**Table 3: Abbreviations**

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

## 1.4 **Overview**

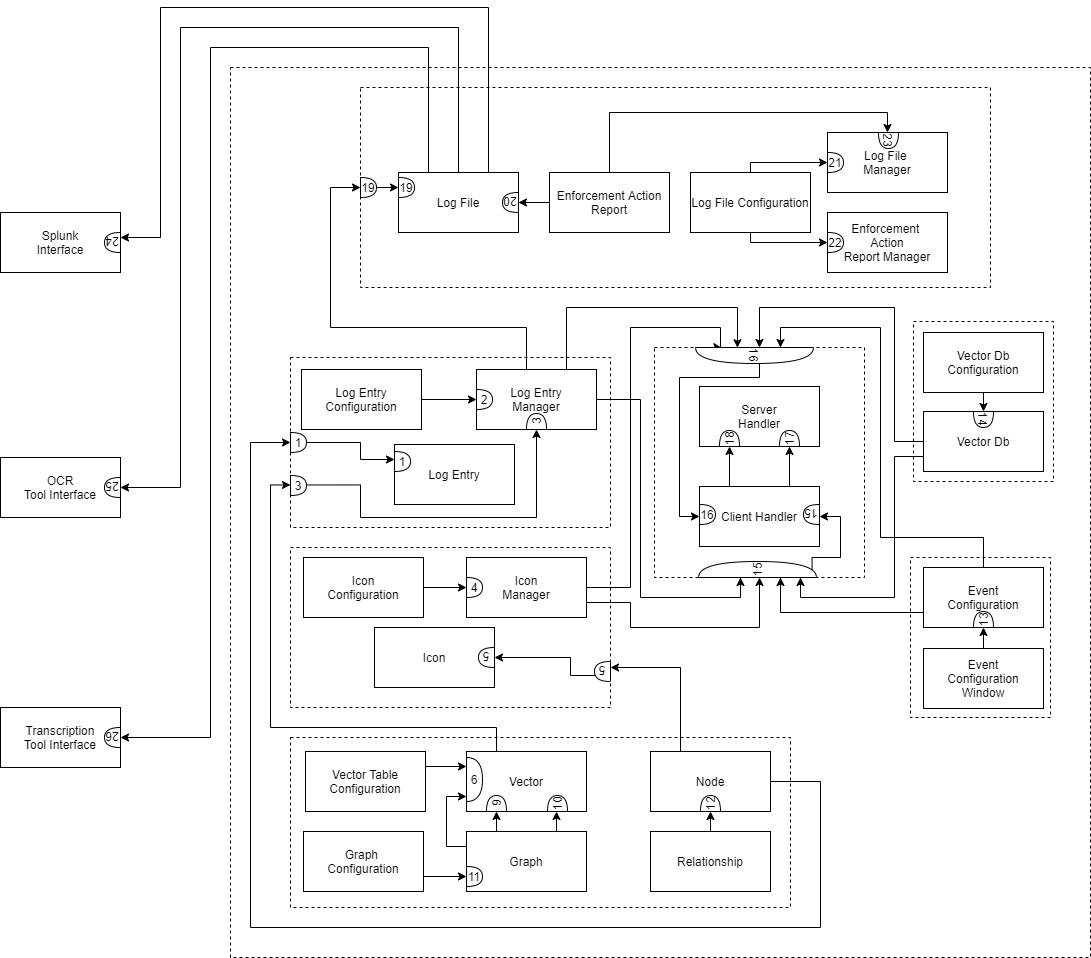
The following section, section 2.0, describes the abstract components of our system, where each class of our system is separated into these components that handle some functionality of our PICK System. We use a system collaboration diagram to visualize these separate components and all the classes within, 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   1. Retrieve Log Entries:   Server: Log Entry Manager   1. Associate Log Entry To Vector   Server Log Entry |
| **Section Discussed:** | Section 3 |

|  |  |
| --- | --- |
| **Component Name:** | Icon Component |
| **Component Purpose:** | Display, manage, and maintain information about icons |
| **Classes:** | Icon, Icon Manager, Icon Configuration |
| **Contracts:** | 1. Retrieve Icon Image 2. Retrieve Icons |
| **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, VectorManager, Node, Relationship, Graph, Vector Configuration, Graph Configuration |
| **Contracts:** | 1. Retrieve Vector Information   Server: Vector Manager   1. Modify Node Information   Server: Vector   1. Create New Relationship   Server: Vector   1. Retrieve Graph Information   Server: Graph Configuration |
| **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:** | 1. Validate Log File 2. Ingest Log File 3. Retrieve Enforcement Action Report 4. Retrieve Log File 5. Modify Log File |
| **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:** | 1. Retrieve Event Configuration Information |
| **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:** | 1. Retrieve Vectors |
| **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:** | 1. Send Store Data Request   Server: Client Handler   1. Send Get Data Request   Server: Client Handler   1. Handle Store Data Request   Server: Server Handler   1. 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:** | 1. 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:** | 1. 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:** | 1. 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 |  |

### 3.1.1.Retreive 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. |

## 3.2 Log Entry Configuration

|  |  |
| --- | --- |
| **Class Name**: Log Entry Configuration | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities**   1. Display Log Entry Manager information 2. Modify Log Entry manager information | |

## 3.3. Log Entry Manager

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

### 3.3.1. Retrieve Log Entries

**Contract Name:** Retrieve Log Entries

**Contract ID:** 2

**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. |

### 3.3.2. Associate Log Entry to Vector

**Contract Name:** Associate log entry to vector

**Contract ID:** 3

**Contract Description:** Associates a provided log entry to a vector

**Protocol:**

|  |
| --- |
| **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 **Subclass**: N/A  **Description**: A component of the system to be produced that will allow the analysts to define elements of the assessment. |
| **Private Responsibilities**:   1. Can store all event configuration information in a system datastore 2. Can retrieve all event configuration from a system datastore 3. Can validate directory structure 4. Can validate directory existence 5. Can perform data cleansing 6. Can perform data ingestion   **Public Responsibilities:**   1. Knows the name of the event 2. Knows the start timestamp of an event 3. Knows the end timestamp of an event 4. Knows the path to the root directory of an event 5. Knows the name of the red team folder 6. Knows the name of the blue team folder 7. Knows the name of the white team folder 8. Knows whether or not the current user is the lead 9. Knows the IP address of the lead 10. Knows the number of connections established to the lead |

### 4.1.1.Retreive Event Configuration Information

**Contract Name:** Retrieve Event Configuration Information

**Contract ID:** 13

**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 name is not null |
| **Post-Condition:** None |
| **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 start time is not null |
| **Post-Condition:** None |
| **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 end time is not null |
| **Post-Condition:** None |
| **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:** Root path is not null |
| **Post-Condition:** The system expects three sub directories in that specified root path; red, blue, and white |
| **Purpose:** Returns the root path to the user |

### 

|  |
| --- |
| **Responsibility:** Knows the name of the red team folder |
| **Method Signature:** def getRedTeamDirectory() |
| **Pre-Conditions:** The root path has been initialized |
| **Post-Condition:** The system knows where the red team log files are located |
| **Purpose:** Returns the path for the red team’s log files |

### 

|  |
| --- |
| **Responsibility:** Knows the name of the blue team folder |
| **Method Signature:** def |
| **Pre-Conditions:** The root path has been initialized |
| **Post-Condition:** The system knows where the blue team log files are located |
| **Purpose:** Returns the path for the blue team’s log files |

### 

|  |
| --- |
| **Responsibility:** Knows the name of the white team folder |
| **Method Signature:** def |
| **Pre-Conditions:** The root path has been initialized |
| **Post-Condition:** The system knows where the white team log files are located |
| **Purpose:** Returns the path for the white team’s log files |

### 

|  |
| --- |
| **Responsibility:** Knows whether or not the current user is the lead |
| **Method Signature:** def isLead(ipAddress) |
| **Pre-Conditions:** Lead machine has been set |
| **Post-Condition:** None |
| **Purpose:** Determines if the given IP address belongs to the lead machine |

### 

|  |
| --- |
| **Responsibility:** Knows the IP address of the lead |
| **Method Signature:** def getLeadIP() |
| **Pre-Conditions:** Lead machine has been set |
| **Post-Condition:** None |
| **Purpose:** Returns the IP address of the lead machine |

### 

|  |
| --- |
| **Responsibility:** Knows the number of connections established to the lead |
| **Method Signature:** def getNumConnections(ipAddress) |
| **Pre-Conditions:** Lead machine has been set |
| **Post-Condition:** None |
| **Purpose:** Lets the system know how many machines are connected to the lead |

### 

## 4.2 Event Configuration Window

|  |
| --- |
| **Class Name:** Event Configuration Window |
| **Superclass**: N/A **Subclass**: N/A  **Description**: Displays event configuration information. |
| **Private Responsibilities**:   1. Can display event configuration information 2. Can modify the information in an event configuration |

## 

# 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 **Subclass**: Lead Vector Database, User Vector Database  **Description**: Creates, deletes, or modifies vectors. |
| **Private Responsibilities**:   1. Can store vectors in a system or local datastore 2. Can retrieve vectors from a system or local datastore 3. Can create a new vector 4. Can modify an existing vector 5. Can delete an existing vector   **Public Responsibilities:**   1. Knows a list of vectors |

## 

### 5.1.1.Retreive Vectors

**Contract Name:** Retrieve Vectors.

**Contract ID:** 14

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

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows a list of vectors |
| **Method Signature:** def |
| **Pre-Conditions:** None |
| **Post-Condition:** The vector has been stored and is now available for the user to select it in the drop down menu for both the tabular and graphical views. |
| **Purpose:** Displays all the vectors in a drop down menu for the user to select |

### 

## 5.2. Vector Db Configuration

|  |
| --- |
| **Class Name:** Vector DB Configuration |
| **Superclass**: N/A **Subclass**: N/A  **Description**: Displays Vector Db information.. |
| **Private Responsibilities**:   1. Can display vector database information 2. Can modify vector database information |

## 

# 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:** 1. Send Store Data Request | |
| **Responsibilities** | **Collaborations** |
| 1. Process store data request | Server Handler (8) |
| **Contract:** 2. Send Get Data Request | |
| **Responsibilities** | **Collaborations** |
| 1. Process get data request from | Server Handler (9) |

### 6.1.1. Send Store Data Request

**Contract Name:** Send Store Data Request

**Contract ID:** 15

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

**Protocol:**

|  |
| --- |
| **Responsibility:** Process store data request |
| **Method Signature:** def sendStoreDataRequest(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 a store data request 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:** 16

**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:** 1. Handle Store Data Request | |
| **Responsibilities** | **Collaborations** |
| 1. Store received data to system datastore. |  |
| **Contract:** 2. Handle Get Data Request | |
| **Responsibilities** | **Collaborations** |
| 1. Retrieve data from a system datastore |  |

### 6.2.1. Handle Store Data Request

**Contract Name:** Handle Store Data Request

**Contract ID:** 17

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

**Protocol:**

|  |
| --- |
| **Responsibility:** Handle Store Data Request |
| **Method Signature:** def handleStoreDataRequest(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 a store data request from a client machine that contains a valid data and protocol. |

### 6.2.2. Handle Get Data Request

**Contract Name:** Handle Get Data Request

**Contract ID:** 18

**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 **Subclass**: Audio Log File, Video Log File, Image Log File, PDF Log File  **Description**: The source log files that the analysts will provide to the system to be cleansed and ingested. |
| **Private Responsibilities**:   1. Knows the file name of the log file 2. Knows the date and time it was most recently updated 3. Knows if the file has been cleansed 4. Knows if the file has been validated, and if so, whether it was successful or not 5. Knows if the file has been ingested into the system database 6. Knows if a user accepts the log file into the system in the event it’s invalid 7. Can cleanse a textual log file   **Public Responsibilities**:   1. Can validate a textual log file 2. Can ingest a textual log file |

## 

### 7.1.1. Validate Log File

**Contract Name:** Validate Log File

**Contract ID:** 20

**Contract Description:** Validates a log file and returns an error if any exists in the log file.

**Protocol:**

|  |
| --- |
| **Responsibility:** Validate Log file |
| **Method Signature:** def validateLogFile(logFile) |
| **Pre-Conditions:** Log file exists |
| **Post-Condition:** The log file becomes validated |
| **Purpose:** To ensure each log file meets the minimum requirements needed to be a valid log file set by the clients. |

### 7.1.2. Ingest Log File

**Contract Name:** Ingest Log File

**Contract ID:** 19

**Contract Description:** Ingests a log file and returns a list of log entries.

**Protocol:**

|  |
| --- |
| **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:** Allows the log entries from a team’s log files to be used throughout the system |

## 7.2 Enforcement Action Report

|  |
| --- |
| **Class Name:** Enforcement Action Report |
| **Superclass**: N/A **Subclass**: N/A  **Description**: A report to be generated when a log file cannot be validated and shows any inaccurate data within that log file. |
| **Private Responsibilities**:   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. Can accept invalid log files as valid 4. Can reject invalid log files |

## 7.3 Enforcement Action Report Manager

|  |
| --- |
| **Class Name:** Enforcement Action Report Manager |
| **Superclass**: N/A **Subclass**: N/A  **Description**: Creates, modifies, and deletes enforcement action reports. |
| **Private Responsibilities**:   1. Can create a new enforcement action report 2. Can modify an existing enforcement action report 3. Can store enforcement action reports in a local data store 4. Can retrieve enforcement action reports from a local datastore 5. Can delete an existing enforcement action report   **Public Responsibilities**:   1. Knows all enforcement action reports created |

## 

### 7.3.1. Retrieve Enforcement Action Report

**Contract Name:** Retrieve Enforcement Action Report

**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:** None |
| **Post-Condition:** All Enforcement Action Reports are returned. |
| **Purpose:** Retrieve and return all Enforcement Action Reports. |

## 7.4 Log File Manager

|  |
| --- |
| **Class Name:** Log File Manager |
| **Superclass**: N/A **Subclass**: N/A  **Description**: Creates, deletes, and modifies log files. |
| **Private Responsibilities**:   1. Can create a new log file 2. Can store log files in a local data store 3. Can retrieve log files from a local data store 4. Can delete an existing log file   **Public Responsibilities**:   1. Knows all log files created 2. Can modify an existing log file |

## 

### 7.4.1. Retrieve Log Files

**Contract Name:** Retrieve Log Files

**Contract ID:** 21

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

**Protocol:**

|  |
| --- |
| **Responsibility:** Retrieve Log files |
| **Method Signature:** def getLogFiles() |
| **Pre-Conditions:** Log files must be validated. |
| **Post-Condition:** Return the log files requested so they may be displayed. |
| **Purpose:** Retrieve and return all log files. |

### 7.4.2. Modify Log File

**Contract Name:** Modify Log File

**Contract ID:** 23

**Contract Description:** Modifies a log file for a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Modify Log File |
| **Method Signature:** def editLogFile(logFile) |
| **Pre-Conditions:** Log file must exist and be validated. |
| **Post-Condition:** Log file is updated and saved |
| **Purpose:** Allows the client to select a log file and modify it. |

## 7.5. Log File Configuration

|  |
| --- |
| **Class Name:** Log File Configuration |
| **Superclass**: N/A **Subclass**: N/A  **Description**: Displays log file information to users. |
| **Private Responsibilities**:   1. Can display Log File Manager information 2. Can display Enforcement Action Report Manager Information 3. Can modify Enforcement Action Report Manager information |

## 

# 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. Knows the icon name of the icon 2. Knows the file path of the icon image | |
| **Contract:** 5. Retrieve Icon Image | |
| **Responsibilities** | **Collaborations** |
| 1. Knows the image of the icon 2. Knows the pixel map for an icon image | Node(6) |

### 8.1.1. Retrieve Icon Image

**Contract Name:** Retrieve Icon Image

**Contract ID:** 5

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

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the image of the icon |
| **Method Signature:**   * Method Name: def getPixmapByteArray(self) * Input/Output Types: N/A, no input or output. |
| **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 image |
| **Method Signature:**   * Method Name: def getGraphImage(self) * Input Type: N/A * Output Type: Image File |
| **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. Can create new icons 2. Can delete an existing icon 3. Can modify an existing icon 4. Can store icons in a system datastore 5. Can retrieve icons from a system datastore | |
| **Contract:** 5. Retrieve Icons | |
| **Responsibilities** | **Collaborations** |
| 1. Knows a list of all icons created | Client Handler(6) |

### 8.2.1. Retrieve Icons

**Contract Name:** Retrieve Icons

**Contract ID:** 4

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

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows a list of all icons created |
| **Method Signature:**   * Method Name: def retrieveIcons(self) * Input Type: N/A * Output Type: List of objects of type Icon. |
| **Pre-Conditions:** None |
| **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. |

## 8.3. Icon Configuration

|  |  |
| --- | --- |
| **Class Name**: Icon Configuration | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   1. Can display Icon Manager information 2. Can modify Icon Manager information | |

# 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 **Subclass**: N/A  **Description**: Collection of significant log entries that describe some adversarial event or story between the blue and red team. |
| **Private Responsibilities**:   1. Knows the user-provided name of the vector. 2. Knows the user-provided description of the vector. 3. Knows the list of log entries that have been associated with the vector. 4. Can create a new node. 5. Can modify an existing node. 6. Can delete an existing node. 7. Can create a new relationship. 8. Can modify an existing relationship. 9. Can delete an existing relationship. 10. Can export the vector information in tabular format.   **Public Responsibilities**:   1. Knows the list of nodes that make up the vector 2. Knows the list of relationships between nodes in the vector. 3. Knows the user-provided name of the vector 4. Knows the user-provided description of the vector 5. Can modify an existing node 6. Can create a new relationship |

## 

### 9.1.1. Retrieve Vector Information

**Contract Name:** Retrieve Vector Information

**Contract ID:** 6

**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 log entries that have been associated with the vector. |
| **Method Signature:** def getAssociatedLogEntries() |
| **Pre-Conditions:** Vector has been initialized. |
| **Post-Condition:** A list of log entries associated to the vector is returned to the client. No vector information is changed in the process. |
| **Purpose:** Retrieve and return a list of the vector’s associated log entries to a client. |

### 9.1.2. Modify Node Information

**Contract Name:** Modify Node Information

**Contract ID:** 9

**Contract Description:** Modifies node information for a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Can create a new node. |
| **Method Signature:** def addSignificantEventFromLogEntry(logEntry) |
| **Pre-Conditions:** Log Entry used to create the node 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:** Can modify an existing node. |
| **Method Signature:** def updateLogEntry(logEntry) |
| **Pre-Conditions:** Log Entry passed to the method has been initialized, the Log Entry ID matches to the node. |
| **Post-Condition:** Node is updated to reflect user-defined modifications. |
| **Purpose:** Modifies a node in the vector using the ID of the log entry it represents. |

|  |
| --- |
| **Responsibility:** Can delete an existing node. |
| **Method Signature:** def removeSignificantEventByLogEntry(logEntryId) |
| **Pre-Conditions:** Log Entry passed to the method has been initialized, the Log Entry ID matches to the node. |
| **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. Create New Relationship

**Contract Name:** Create New Relationship

**Contract ID:** 10

**Contract Description:** Creates a new relationship between nodes for a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Can create a new relationship. |
| **Method Signature:** def addNewRelationship(sourceId, destId) |
| **Pre-Conditions:** The source node and destination node have been initialized, valid node IDs are provided. |
| **Post-Condition:** A new relationship between two given nodes is created and added to the vector. |
| **Purpose:** Creates a new relationship between two nodes and adds relationship to the vector. |

## 9.2. Graph

|  |
| --- |
| **Class Name:** Graph |
| **Superclass**: N/A **Subclass**: N/A  **Description**: The visual representation of a vector and its related nodes and connectors. |
| **Private Responsibilities**:   1. Can edit the positions of the nodes. 2. Can associate nodes together as relationships 3. Can export the nodes and relationships as an image.   **Public Responsibilities:**   1. Knows the list of nodes that make up the graph 2. Knows the list of relationships that make up the graph |

## 

### 9.2.1. Retrieve Graph Information

**Contract Name:** Retrieve Graph Information

**Contract ID:** 11

**Contract Description:** Retrieve graph information (list of nodes and relationships that make up a graph) for a client.

**Protocol:**

|  |
| --- |
| **Responsibility:** Knows the list of nodes and relationships that make up the graph. |
| **Method Signature:** def getGraphInformation() |
| **Pre-Conditions:** Graph has been initialized. |
| **Post-Condition:** The list of nodes and relationships that make up the graph is returned to the client. No graph information is changed in the process. |
| **Purpose:** Retrieve and return the list of nodes and relationships that make up the graph to a client. |

## 9.3. Vector Table Configuration

|  |
| --- |
| **Class Name:** Vector Table Configuration |
| **Superclass**: N/A **Subclass**: N/A  **Description**: Displays vector information to users. |
| **Private Responsibilities**:   1. Can display Vector Information 2. Can modify Vector Information |

## 

## 9.4. Graph Configuration

|  |
| --- |
| **Class Name: Graph Configuration** |
| **Superclass**: N/A **Subclass**: N/A  **Description**: Display Graph to users. |
| **Private Responsibilities**:   1. Can display Graph information 2. Can modify Graph information |

## 9.5. Node

|  |
| --- |
| **Class Name:** Node |
| **Superclass**: N/A **Subclass**: N/A  **Description**: A significant log entry in a vector. |
| **Private Responsibilities**:   1. Knows the node name 2. Knows the node timestamp 3. Knows the node description 4. Knows the log entry reference 5. Knows the log entry information of the log entry reference 6. Knows the node’s icon information 7. Knows the node’s position on the graph 8. Knows the visibility of the node’s ID on the graph 9. Knows the visibility of the node’s name on the graph 10. Knows the visibility of the node’s timestamp on the graph 11. Knows the visibility of the node’s description on the graph 12. Knows the visibility of the node’s log entry reference on the graph 13. Knows the visibility of the node’s event type on a graph 14. Knows the visibility of the node’s icon on a graph 15. Knows the visibility of the node’s source on a graph 16. Knows the node’s visibility on a graph   **Public Responsibilities:**   1. Knows the node ID |

### 9.5.1. Retrieve Node ID

**Contract Name:** Retrieve Node ID

**Contract ID:** 12

**Contract Description:** Retrieves and returns a node ID for a client.

**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. |

# 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:** 1. 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:** 24

**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:** 1. 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:** 25

**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:** 1. 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:** 26

**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. |

**$**