PICK

**Software Design Document**

version 1.4

3/8/2020

**Document Control**

**Approval**

The Guidance Team and the customer shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 3/2/2020 |
| Current Release: | 1.4 |
| Indicator of Last Page in Document: | **%** |
| Date of Last Review: | 3/8/2020 |
| Date of Next Review: | 3/9/2020 |
| Target Date for Next Update: | 3/9/2020 |

**Distribution List**

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

**Guidance Team Members:** Dr. Steve Roach, Jake Lasley

**Customer:** Army Research Lab which consists of Herandy Denisse Vazquez, Baltazar Santaella, Vincent Fonseca, Oscar Perez, and Florencia Larsen.

**Software Team Members:** Zayra Padilla, Erik Macik, Priscilla Mendoza, Jose Lujan, Michael Contreras

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 3/2/2020 | Team | Creation of Document |
| 1.2 | 3/08/2020 | Team | Finished section 2. |
| 1.3 | 3/08/2020 | Team | Finished section 3-6 |
| 1.4 | 3/08/2020 | Team | Finished section 7 |
| 1.5 | 3/29/2020 | Team | Finished Collaboration diagram |
| 1.6 | 3/30/2020 | Team | Added protocols |

Table of Contents

[**DOCUMENT CONTROL II**](#_heading=h.30j0zll)

[Approval ii](#_heading=h.3as4poj)

[Document Change Control ii](#_heading=h.1pxezwc)

[Distribution List ii](#_heading=h.49x2ik5)

[Change Summary ii](#_heading=h.2p2csry)

[**1.**](#_heading=h.147n2zr) **INTRODUCTION 1**

[1.1.](#_heading=h.3o7alnk) Purpose and Intended Audience 1

[1.2.](#_heading=h.23ckvvd) Scope of Product 1

[*1.2.1.*](#_heading=h.ihv636) *Database 1*

[*1.2.2.*](#_heading=h.32hioqz) *Notification Manager 1*

[*1.2.3.*](#_heading=h.1hmsyys) *Unit Conversion 1*

[1.3.](#_heading=h.41mghml) References 1

[1.4.](#_heading=h.17dp8vu) Definitions, Acronyms, and Abbreviations 2

[*1.4.1.*](#_heading=h.2grqrue) *Definitions 2*

[*1.4.2.*](#_heading=h.vx1227) *Acronyms 2*

[*1.4.3.*](#_heading=h.3fwokq0) *Abbreviations 2*

[1.5.](#_heading=h.1v1yuxt) Overview 2

[**2.**](#_heading=h.4f1mdlm) **DECOMPOSITION DESCRIPTION 3**

[2.1.](#_heading=h.2u6wntf) Scope 3

[2.2.](#_heading=h.19c6y18) Use 3

[2.3.](#_heading=h.3tbugp1) Subsystem Description 3

[2.4.](#_heading=h.28h4qwu) Hierarchy Graphs 4

[*2.4.1.*](#_heading=h.nmf14n) *Database Manager Subsystem 4*

[*2.4.2.*](#_heading=h.37m2jsg) *Database Manager (API) 4*

[*2.4.3.*](#_heading=h.1mrcu09) *Unit Conversion 5*

[*2.4.4.*](#_heading=h.46r0co2) *Notification Manager 5*

[**3.**](#_heading=h.2lwamvv) **DEPENDENCY DESCRIPTION 6**

[3.1.](#_heading=h.111kx3o) Scope 6

[3.2.](#_heading=h.3l18frh) Use 6

[3.3.](#_heading=h.206ipza) Collaboration Description 6

[**4.**](#_heading=h.4k668n3) **DETAILED DESIGN 8**

[4.1.](#_heading=h.2zbgiuw) Scope 8

[4.2.](#_heading=h.1egqt2p) Use 8

[4.3.](#_heading=h.3ygebqi) Components 8

[*4.3.1.*](#_heading=h.2dlolyb) *Database Manger 8*

[*Scenario 1: Perform insert, update, select, or delete query. 8*](#_heading=h.sqyw64)

[*Scenario 1: Upload. 9*](#_heading=h.3cqmetx)

[*Scenario 2: Download. 9*](#_heading=h.1rvwp1q)

[*4.3.2.*](#_heading=h.4bvk7pj) *Unit Conversion 9*

[*Scenario 1: Request conversion of units. 9*](#_heading=h.2r0uhxc)

[*4.3.3.*](#_heading=h.1664s55) *Notification mails 10*

[*Scenario 1: Edit data. 10*](#_heading=h.3q5sasy)

[*Scenario 2: Add new weather station. 10*](#_heading=h.25b2l0r)

[4.4.](#_heading=h.qsh70q) Database Schema 11

# **Introduction**

## **Purpose and Intended Audience**

The purpose of the Software Design Document (SDD) is to give the developers a clear description of the design of the system. It will provide a way for software development to go about by giving a thorough understanding of what is to be built and how to build the system. It will list enough detail of all subsystems and components that make up the system and how each responsibility collaborates with other components to fulfill the system. The intended audience of the SDD is Dr. Oscar Perez, Mr. Vincent Fonseca, Ms. Herandy Vazquez, Mr. Baltazar Santaella, Ms. Florencia Larsen and the guidance team.

## **Scope of Product**

The Lethality, Survivability, and HSI Directorate (LSH) recognizes the complexity and the time it takes to analyze the applicable logs, observation notes, and other artifacts gathered from an adversarial assessment from the red, blue, and white teams and generate a report that represents the events that took place during the adversarial assessment. They want a system that would aid their analysts in correlating red team’s activities to blue team’s responses and represent the events that took place during an adversarial assessment graphical.

The University of Texas at El Paso (UTEP) and LSH are collaborating to develop Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK) that will provide the ability to correlate red team’s activities to blue team’s responses and graphically represent the events that took place during an adversarial assessment.

## **References**

There are no references.

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

### **Definitions**

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| Data Cleansing | Data cleansing is the removal of unwanted characters from uncleansed TMUX log file; removal of blank rows from uncleansed excel log file; and removal of blank lines from uncleansed log file. |
| Data Validation | Data validation is the process of inspecting data in the cleansed log files based on predefined data validation rules. |
| Log Entry | Splunk takes the validated log files and convert them into normalized data. The normalized data are called log entries. Users of the system can filter and edit log entries. |
| Significant Log Entry | A log entry selected by the user and associated with a vector. The attributes are the same as for a log entry. The system stores significant log entries. Splunk stores log entries in the normalized data files. |
| Timestamp | Denotes time in hours:minutes, date in month:date:year, and section in am/pm. |
| Significant log entry | Denotes a log entry that is associated to at least one vector. |

### **Acronyms**

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| UTEP | The University of Texas at El Paso |
| PICK | Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System |
| LSH | Lethality, Survivability, and HSI Directorate |
| PMR | Prevent, Mitigate, and Recover |
| IP | Internet Address |
| AA | Adversarial Assessment |

### **Abbreviations**

|  |  |
| --- | --- |
| **TERM** | **DEFINITION** |
| e.g. | For example |
| i.e. | That is |
| TBD | To be determined |

## **Overview**

The SDD is divided into four major sections: Introduction (Section 1), Decomposition Description (Section 2), Detailed Description of Component (Section 3 to 6) and Database (Section 7).

Section 1 includes a brief description of the system, what an SDD is and what purpose it serves, the intended audience and the organization of the document.

Section 2 is made up of three subsections. Section 2.1 provides a Wirfs-Brock Collaboration Diagram. Section 2.2 provides a description of subsystems and components in the diagrams in section 2.1. Section 2.3 describes how the component dependencies will impact development.

Section 3 to 6 is made up of two or more subsections each. Section n.1 provides the component’s description. Section n.2 and so on provides the listed class’s description and its corresponding contract.

Section 7 provides the database schema and layout as well as a brief description of all the different tables in the schema.

# **Decomposition Description**

## **System Collaboration Diagram**

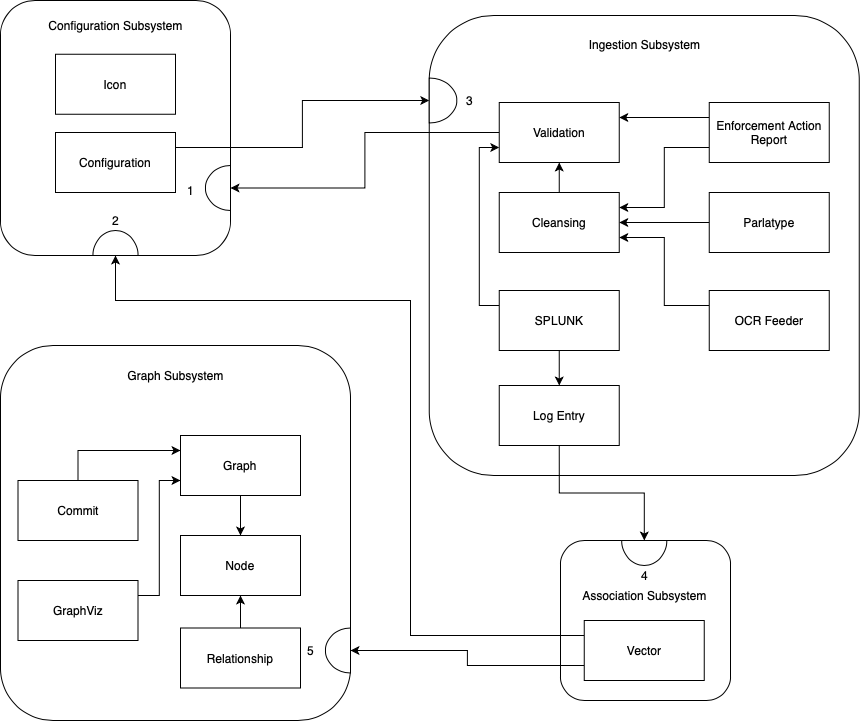


Figure 1: Wirfs-Brock Collaboration Diagram for PICK System

## **Subsystem and Component Descriptions**

Configuration Subsystem (Section 3)

* Purpose - Know an initial setup for the system that includes event, team, directories, icons, and vectors.
* Supported Contracts
  + 1. Provide log files
  + 2. Provide configuration details

Ingestion Subsystem (Section 4)

* Purpose - Responsible for transcribing, cleansing, and validating log files, and sending log files for Splunk ingestion.
* Supported Contracts
  + 3. Turn log files into log entries

Association Subsystem (Section 5)

* Purpose - Responsible for creating the relationships given nodes.
* Supported Contacts
  + 4. Associate log entries to a particular vector

Graph Subsystem (Section 6)

* Purpose - Responsible for creating a graphical representation of a vector and its associations.
* Supported Contracts
  + 5. Create a graphical representation of vector

## **Dependencies**

The system can be run on Linux, Mac OS and Windows 10.

Required components for system to run:

* Python 3
* PyQt5
* MongoDB
* Splunk SDK
* QGraphViz
* OCR Feeder
* Parlatype

# **Detailed Description of** Configuration Subsystem

## Subsystem **Description**

**Subsystem name:** Configuration Subsystem

**Purpose of subsystem:** Know an initial setup for the system that includes event, team, directories, icons, and vectors.

**Classes contained in subsystem:** Configuration, Vector

## **Class Description of** Configurations

|  |  |
| --- | --- |
| **Class Name**: Configurations | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities**   * Obtain files in known directories. * Obtain event details * Obtain team details * Obtain icon details * Update existing icon fields. | |
| **Contract 1:** Provide configuration details | |
| **Responsibilities** | **Collaborations** |
| 1. Know the event’s name. 2. Know the event’s description. 3. Know the event’s start date. 4. Know the event’s start time. 5. Know the event's end date. 6. Know the event's end time. 7. Know the lead host machine where the master vector DB is stored. 8. Know the number of established connections. 9. Know icon names. 10. Know icon sources. 11. Know the directories needed to retrieve log files. 12. Delete selected icons. | N/A |
| **Protocol:** getConfigurations() | |
| **Pre-Condition:** A configuration must exist. | |
| **Post-Condition:** Returns the configuration. | |
| **Description:** Will store all configuration attributes. | |

## **Class Description of** Vector

|  |  |
| --- | --- |
| **Class Name**: Vector | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   1. Know the vector ID. 2. Know the vector name. 3. Know the vector description. 4. Know which log entries are associated with different vectors. 5. Delete existing vectors along with all of its associations. | |
| **Contract 2:** Provide associated log entries | |
| **Responsibilities** | **Collaborations** |
| 1. Associate log entry to given vector. 2. Update vector information in the database based on changed information by the user. | * Node (7) |
| **Protocol:** getVector() | |
| **Pre-Condition:** Associated log entries must exist. | |
| **Post-Condition:** Return associated log entries. | |
| **Description:** Store associated log entries to a selected vector. | |

# Detailed Description of Ingestion Subsystem

## Subsystem Description

**Subsystem name:** Ingestion Subsystem

**Purpose of subsystem:** Responsible for transcribing, cleansing, and validating log files, and sending log files for Splunk ingestion.

**Classes contained in subsystem:** Log Entry, Cleansing, Validation, Enforcement Action Report, Splunk, OCR Feeder, Parlatype

## Class Description of Log Entry

|  |  |
| --- | --- |
| **Class Name**: Log Entry | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:** N/A | |
| **Contract 3:** Provide log entry details | |
| **Responsibilities** | **Collaborations** |
| 1. Know log entry ID. 2. Know time of occurrence 3. Know description 4. Know log file path 5. Know log creator 6. Know the event type 7. Know artifact | Configurations (1) |
| **Protocol:** getLogEntryDetails() | |
| **Pre-Condition:** Have an existing log entry. | |
| **Post-Condition:** Return Log Entry attributes. | |
| **Description:** The method will provide the fields stored in a given log entry. | |

## Class Description of Cleansing

|  |  |
| --- | --- |
| **Class Name**: Cleansing | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   1. Know cleansing status. 2. Know cleansing errors. | |
| **Contract 4:** Cleanse log file according to script. | |
| **Responsibilities** | **Collaborations** |
| 1. Know cleansing script. 2. Run cleansing script. 3. Send log files to Audio Transcription Tool. 4. Send log files to OCR Transcription Tool. | Configurations (1) |
| **Protocol:** cleanse() | |
| **Pre-Condition:** Have existing log files to cleanse. | |
| **Post-Condition:** Return cleansed log files. | |
| **Description:** This method will run a cleansing script and if needed, will send log files to the Audio or OCR Transcription Tool. | |

## Class Description of Validation

|  |  |
| --- | --- |
| **Class Name**: Validation | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   1. Know Validation errors. 2. Know Validation status. | |
| **Contract 5:** Approve/Deny cleansed file | |
| **Responsibilities** | **Collaborations** |
| 1. Validate that the cleansed log file content falls within the start date/time. 2. Validate that the cleansed log file content falls within the end date/time. 3. Validate that the cleansed log file content does not contain any empty rows. 4. Begin SPLUNK ingestion by sending the validated log files. | Cleansing (4)  Splunk (11) |
| **Protocol:** validate() | |
| **Pre-Condition:** Have existing cleansed files. | |
| **Post-Condition:** Return approved or denied validation status. | |
| **Description:** This method will contain validation rules. | |

## Class Description of Enforcement Action Report

|  |  |
| --- | --- |
| **Class Name**: Enforcement Action Report | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   1. References the status of log files and their error descriptions. | |
| **Contract 6:** Provide error descriptions for files that fail ingestion. | |
| **Responsibilities** | **Collaborations** |
| 1. Display log files that could not be ingested. 2. Display error descriptions for log files. | Cleansing (4)  Validation (5) |
| **Protocol:** actionReport() | |
| **Pre-Condition:** Have existing cleansed files with their validation status. | |
| **Post-Condition:** Returns the log files with their error descriptions. | |
| **Description:** This method stores error descriptions for log files. | |

## Class Description of SPLUNK

|  |  |
| --- | --- |
| **Class Name**: Splunk | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   * Store log entries. * Receive log files. | |
| **Contract 11:** Transform log files into log entries. | |
| **Responsibilities** | **Collaborations** |
| 1. Transform log files to log entries. 2. Send log entries. | Validation (5) |
| **Protocol:** startIngestion() | |
| **Pre-Condition:** Have existing validated log files. | |
| **Post-Condition:** Return log entries. | |
| **Description:** This method will start ingestion by sending log files to Splunk so that Splunk can transform them into log entries. | |

## Class Description of OCR Feeder

|  |  |
| --- | --- |
| **Class Name**: OCRFeeder | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   * Export as selected format. | |
| **Contract 14:** Create log files from pdf files and images. | |
| **Responsibilities** | **Collaborations** |
| 1. Transcribe log files received. 2. Send transcribed files back. | Cleansing (4) |
| **Protocol:** ocr() | |
| **Pre-Condition:** Have a file that needs to be transcribed. | |
| **Post-Condition:** Return transcribed file. | |
| **Description:** This method will send selected files to the OCR Feeder tool for text transcription. | |

## Class Description of Parlatype

|  |  |
| --- | --- |
| **Class Name**: Parlatype | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:** N/A | |
| **Contract 15:** Create log files from pdf files and images. | |
| **Responsibilities** | **Collaborations** |
| 1. Transcribe mp3 files to text files. 2. Send transcribed files back | Cleansing (4) |
| **Protocol:** parlatype() | |
| **Pre-Condition:** Have an audio file. | |
| **Post-Condition:** Return transcribed file. | |
| **Description:** This method will send audio files to Parlatype for text transcription. | |

# Detailed Description of Association Subsystem

## Subsystem Description

**Subsystem name:** Association Subsystem

**Purpose of subsystem:** Responsible for creating the relationships given nodes.

**Classes contained in subsystem:** Log Entry, Vector

## Class Description of Vector

|  |  |
| --- | --- |
| **Class Name**: Vector | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   1. Know the vector ID. 2. Know the vector name. 3. Know the vector description. 4. Know which log entries are associated with different vectors. 5. Delete existing vectors along with all of its associations. | |
| **Contract 2:** Provide associated log entries | |
| **Responsibilities** | **Collaborations** |
| 1. Associate log entry to given vector. 2. Update vector. | * Node (7) |
| **Protocol:** getVector() | |
| **Pre-Condition:** Associated log entries must exist. | |
| **Post-Condition:** Return associated log entries. | |
| **Description:** Store associated log entries to a selected vector. | |

# Detailed Description of Graph Subsystem

## Component Description

**Subsystem name:** Graph Subsystem

**Purpose of subsystem:** Responsible for creating a graphical representation of a vector and its associations.

**Classes contained in subsystem:** Graph, Node, Relationship, Commit, GraphViz

## Class Description of Graph

|  |  |
| --- | --- |
| **Class Name**: Graph | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   1. Display vector name. 2. Display vector description 3. Display graph in vertical position. 4. Display graph in horizontal position. 5. Zoom in and zoom out into the graph 6. Display the graph in the selected time interval units. 7. Know relationships for each vector. 8. Know nodes for each vector. 9. Add blank node to the graph. 10. Remove node from the graph. 11. Edit node. 12. Commit. | |
| **Contract 8:** Display/Hide Node details | |
| **Responsibilities** | **Collaborations** |
| 1. Display node id. 2. Display node name. 3. Display node time of occurrence. 4. Display node long description. 5. Display log entry reference. 6. Display log creator. 7. Display node event type. 8. Display all node properties. 9. Hide node id. 10. Hide node name. 11. Hide node time of occurrence. 12. Hide node long description. 13. Hide log entry reference. 14. Hide log creator. 15. Hide node event type. 16. Hide node artifact. 17. Hide all node properties. | Node (7)  Vector (2) |
| **Protocol:** nodeVisibility() | |
| **Pre-Condition:** Have an existing Node. | |
| **Post-Condition:** Return Node visibility. | |
| **Description:** This method will know which of the Node attributes to display. | |

## Class Description of Node

|  |  |
| --- | --- |
| **Class Name**: Node | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:** N/A | |
| **Contract 7:** Provide Node details | |
| **Responsibilities** | **Collaborations** |
| 1. Know node id. 2. Know node name. 3. Know the log entry it will reference 4. Know the vector it belongs to. 5. Know Icon id. | Log Entry (3) |
| **Protocol:** getNode() | |
| **Pre-Condition:** Have existing Nodes. | |
| **Post-Condition:** Return Node details | |
| **Description:** This method will store Node attributes. | |

## Class Description of Relationship

|  |  |
| --- | --- |
| **Class Name**: Relationships | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   * Know relationship identity number. * Know label for the relationship | |
| **Contract 9:** Provide relationship between 2 nodes. | |
| **Responsibilities** | **Collaborations** |
| 1. Know parent node 2. Know child node 3. Provide associations. | Node (7) |
| **Protocol:** getRelationship() | |
| **Pre-Condition:** Have existing relationships. | |
| **Post-Condition:** Return relationship. | |
| **Description:** This method will store the Relationship attributes. | |

## Class Description of Commit

|  |  |
| --- | --- |
| **Class Name**: Commit | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:**   * Know changes to be committed * Commit. | |
| **Contract 10:** Save changes. | |
| **Responsibilities** | **Collaborations** |
| 1. Confirm all changes that will be updated within the previous version. | Graph (8) |
| **Protocol:** commit() | |
| **Pre-Condition:** Know changes to be committed. | |
| **Post-Condition:** Commit | |
| **Description:** This method will confirm and commit the analyst’s changes. | |

## Class Description of GraphViz

|  |  |
| --- | --- |
| **Class Name**: GraphViz | |
| **Superclass**: N/A | |
| **Subclasses**: N/A | |
| **Private Responsibilities:** N/A | |
| **Contract 12:** Create graph. | |
| **Responsibilities** | **Collaborations** |
| 1. Display nodes. 2. Display connectors 3. Display icons. 4. Export as selected format. | Graph (8) |
| **Protocol:** graphViz() | |
| **Pre-Condition:** Have the data needed for the graph. | |
| **Post-Condition:** Return a graph. | |
| **Description:** This method will send data to GraphViz to create a graph will the data provided. | |
| **Contract 13:** Export graph. | |
| **Responsibilities** | **Collaborations** |
| 1. Export a graph in a selected format. | Graph (8) |
| **Protocol:** export() | |
| **Pre-Condition:** Have an existing graph to export. | |
| **Post-Condition:** Export. | |
| **Description:** This method will export the given graph. | |

# **Database**

## **Database Schema**

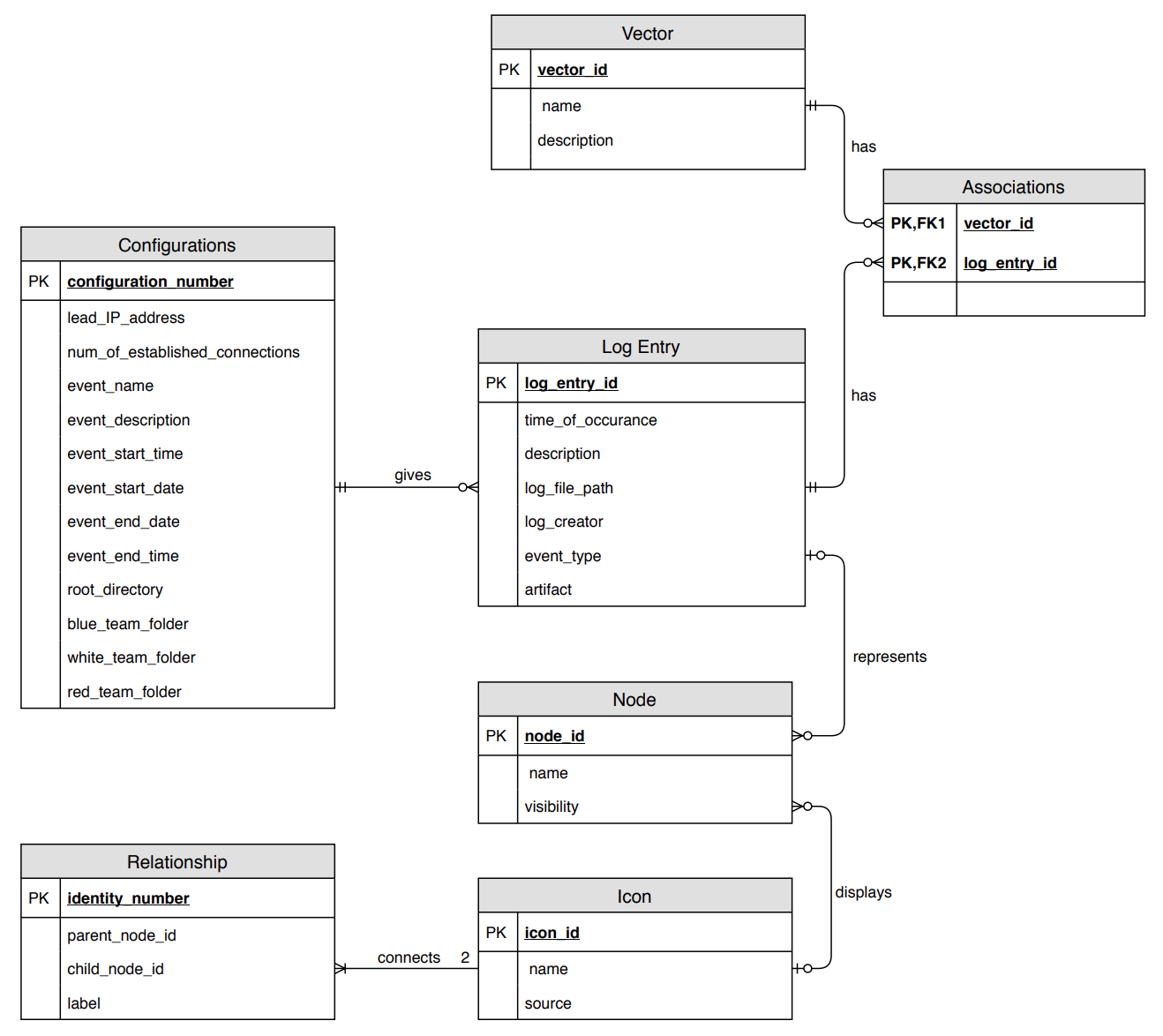


Figure 2: Entity-Relationship Diagram with Crow’s Foot notation

Configurations

The configuration table is an abstraction for all the different types of configurations. Each event will have its own corresponding configuration data which will be stored in this table and retrieved through a unique configuration number.

Log Entry

The Log Entry table will only store log entries and their corresponding attributes if they are associated with any existing vector. These log entries will be retrieved from splunk and stored the moment they are associated to any existing vector.

Vector

The vector table will store different vectors created within the system by any user. Each vector has associations to different log entries.

Associations

The associations table will store the associations between log entries and vectors by storing the primary keys of both the log entry primary key and vector primary key.

Node

The node table will store all existing nodes. They are needed to represent a log entry in a graphical view.

Icon

The icon table will store all uploaded icons that will be used to visually display different images for each node.

Relationship

The relationship table will store all relationships that exist between different nodes.

**%**