We Showed Up

Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK) Software Design Document

2.1

03/21/20

<team>

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

Initial Release:	1.0
Current Release:	2.1
Indicator of Last Page in Document:	*
Date of Last Review:	03/06/2020
Date of Next Review:	03/31/2020
Target Date for Next Update:	03/31/2020

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. Roach
- Jake Lasley

Customer:
- Dr. Oscar Perez
- Vincent Fonseca
- Herandy Denisse Vazquez
- Baltazar Santaella
- Florencia Larsen
- Erick De Nava

Software Team Members:
- Daniela Garcia
- Ricardo Alvarez
- Diego Rincon
- Mathew Iglesias
- Jessica Redekop

Change Summary

The following table details changes made between versions of this document

Version	Date	Modifier	Description
1.0	03/06/2020	Jessica Redekop	Section 1: Introduction
1.1	03/08/2020	Jessica Redekop	Collaboration Diagrams
1.2	03/08/2020	Matthew Iglesias	Section 4: Database Schema Diagram
1.3	03/08/2020	Ricardo Alvarez	Section 3, Initial description of classes
1.4	03/08/2020	Diego Rincon	Section 2.2 & 2.3
1.5	03/09/2020	Daniela Garcia	CRC cards and Section 3 tables
1.6	03/09/2020	Ricardo Alvarez	Initial version of contract #1
1.7	03/09/2020	Jessica Redekop	Collaboration Graphs
1.8	03/29/2020	Jessica Redekop	Section 3.1: Contracts for Ingestion Subsystem
1.9	03/30/2020	Ricardo Alvarez	Section 3.1: Contracts pt.2 and Subsystem

			UML
2.0	03/30/2020	Matthew Iglesias	Section 3.3
2.1	03/30/2020	Daniela Garcia	Section 3.2: contracts for Graphing subsystem

Table of Contents

Deleted: DOCUMENT CONTROL→ II¶
APPROVAL→ II¶
DOCUMENT CHANGE CONTROL→ II¶
DISTRIBUTION LIST→ II¶
CHANGE SUMMARY→ II¶ ... [1]

Deleted: ¶ ... [2]
Deleted: ¶ ... [3]
Deleted: ¶ ... [4]
Deleted: ¶ ... [5]
Deleted: ¶ ... [6]
Deleted: ¶ ... [7]
Deleted: ¶ ... [8]
Deleted: ¶ ... [9]
Deleted: ¶ ... [10]
Deleted: ¶ ... [11]
Deleted: ¶ ... [12]
Deleted: ¶ ... [13]
Deleted: ¶ ... [14]
Deleted: ¶ ... [15]
Deleted: ¶ ... [16]
Deleted: ¶ ... [17]
Deleted: ¶ ... [18]
Deleted: ¶ ... [19]
Deleted: ¶ ... [20]
Deleted: ¶ ... [21]
Deleted: ¶ ... [22]
Deleted: ¶ ... [23]
Deleted: ¶ ... [24]
Deleted: ¶ ... [25]
Deleted: ¶ ... [26]
Deleted: ¶ ... [27]
Deleted: ¶ ... [28]
Deleted: ¶ ... [29]
Deleted: ¶ ... [30]
Deleted: ¶ ... [31]
Deleted: ¶ ... [32]
Deleted: ¶ ... [33]
Deleted: ¶ ... [34]
Deleted: ¶ ... [35]
Deleted: ¶ ... [36]
Deleted: ¶ ... [37]
Deleted: ¶ ... [38]
Deleted: ¶ ... [39]
Deleted: ¶

1. Introduction

1.1. Purpose and Intended Audience

The purpose of creating the Software Design Document (SDD) is to give the customer a clean and precise description of the system design of the Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK). The SDD divides the system design specifications into the following three parts: The decomposition of the system, a collaboration description, and the detailed design of the system's subcomponents.

The intended audience of the SDD is Dr. Oscar Perez, Mr. Vincent Fonseca, Ms. Herandy Vazquez, Mr. Baltazar Santaella, Ms. Florencia Larsen, and the Software Engineering teams. This document serves as an agreement between both parties regarding the system to be developed.

1.2. 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 presents 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 graphically.

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. [1]

1.3. References

[1] R. Alvarez, D. Garcia, M. Iglesias, J. Redekop, D. Rincon, "PICK Responsibilities, and Collaborations."

[2] S. Roach, and E. T. Ramirez, "PICK Software Requirements and Specification."

Deleted: [1] SRS_v7¶

1.4. Definitions, Acronyms, and Abbreviations

1.4.1. Definitions

The definitions in this section are given in the context of the product being developed. This intention is to assist the user in their understanding of the document.

Table 1: Definitions

TERM	DEFINITION
Cleansing	The removal of unwanted characters from uncleansed TMUX log files, blank rows from uncleansed excel log files, and blank lines from uncleansed log files.
Log Entry	Log files feed into and normalized through SPLUNK. Log entries can be filtered and edited by users of the system.
Timestamp	Denotes date, time, and section in the following format: month:date:year hours:minutes, and section in am/pm.

1.4.2. Acronyms

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

Table 2: Acronyms

TERM	DEFINITION
SDD	Software Design Documentation
SRS	System Requirements Specification
UTEP	The University of Texas at El Paso
PICK	Prevent, Mitigate, and Recover (PMR) Insight
LSH	Lethality, Survivability, and HSI Directorate
PMR	Prevent, Mitigate, and Recover
IP	Internet Address
AA	Adversarial Assessment
UI	User Interface
VCS	Version Control Subsystem

1.4.3. Abbreviations

This section provides a list of used abbreviations and their associated definitions.

Table 3: Abbreviations

TERM	DEFINITION

1.5. Overview

The SDD is divided into four major sections: Introduction (Section 1), Decomposition Description (Section 2), Dependency Description (Section 3), and Detailed Design (Section 4).

Section 1 includes five sections. Section 1.1 describes the purpose and intended audience. Section 1.2 describes the scope of the system. Section 1.3 contains the references used throughout the document. Section 1.4 defines the definitions, acronyms, and abbreviations used throughout the document. Section 1.5 is this overview of the SDD.

Section 2 includes four sections. Section 2.1 defines the scope of the decomposition descriptions of the system. Section 2.2 describes the use of the decompositions of the system. Section 2.3 has the subsystem description. Section 2.4 contains the hierarchy graphs of the system.

Section 3 includes three sections. Sections 3.1 describes the scope of the dependency descriptions of the system. Section 3.2 describes the use of the dependencies of the system. Section 3.3 contains the collaboration descriptions between the system.

Section 4 includes four sections. Section 4.1 defines the scope of the detailed design of the system components. Section 4.2 describes the use of the detailed design for the system. Section 4.3 describes the components of the system. Section 4.4 contains the database schemas for the system.

2. Decomposition Description

The following section depicts the system component collaboration diagram, identifies one subsystem and describes its components to identify which entity is responsible for specific functions, and finally describes how the component dependencies will impact development.

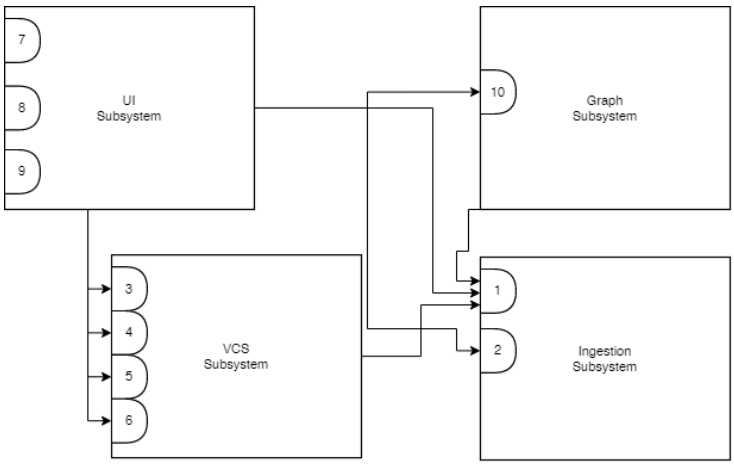
2.1. System Collaboration Diagram

<< Provide a UML Component Diagram or a Wirffs-Brock Collaboration Diagram. If this is a subsystem or part of a larger system, show the collaboration or component diagram for the entire system in a separate diagram first. Show the major components or subsystems in this system and indicate collaborations between components. If useful, show the UML class diagram that indicates class hierarchies.>>

<< Provide a description of the way the system has been structured and the major divisions between the design entities. Subsystems and classes are referred as design entities >>
The system contains all user interface (UI) items

2.1.1. UML System Component Diagram

Diagram 1: UML System Component Diagram



2.2. Subsystem and Component Descriptions

2.2.1. Ingestion

The ingestion subsystem cleanses and validates the log files before normalization, transcribes audio and video into text files, and feeds the validated log files into SPLUNK to convert them into log entries. The subsystem is composed of the classes IngestionController, EventConfiguration, LogFile, EnforcementActionReport, SplunkFaçade, TranscriberInterface, and LogEntry, and consists of the following contracts:

- Contract 1: Provide log entries
- Contract 2: Provide the list of the errors during the ingestion process pertaining to each log file

Deleted: files

Deleted: <#>Contract 4: Provide SPLUNK log entries¶

A detailed description of the ingestion subsystem will be discussed in section 3.

2.2.2. VCS

The version control subsystem (VCS) tracks the changes in the vectors in the analysts' vector database, allows analysts to push and pull vector changes into the database, and allows the lead analyst to control the changes pushed to the database. The subsystem is composed of the classes VersionControl, Vector, Vector Database, and EventConfiguration, and consists of the following contracts:

- Contract 3: Track changes in vector database
- Contract 4: Push changes to the vector database
- Contract 5: Pull changes from the vector database
- Contract 6: Control pushes to the vector database

2.2.3. UI

The UI subsystem updates the information displayed in the log entry, and vector tables, and controls the information displayed in the tables through filters. The subsystem is composed of the classes TableManager, Vector, UIFunctionality, LogEntry, FilterFunctionality, and UIView, and the following contracts:

- Contract 7: Display log entry data in the vector table
- Contract 8: Display filtered entries in tables
- Contract 9: Edit data models from user input

Deleted: <#>Table¶

Deleted: table

2.2.4. Graph

The graph subsystem updates the information of the nodes displayed in the graph of the vector and controls the elements of graph UI. The subsystem is composed of the classes GraphFunctionality, QTGraph, Graph, Node and Node, and the following contracts:

- Contract 10: Interact with QTGraph API to create a graphical representation of nodes and relationships

Deleted: Node, UIView,

Deleted: and

Deleted: UIFunctionality

Deleted: Get graph UI elements ¶
<#>Contract 14: Get graph vector data ¶
<#>Contract 15: Display graph vector data in graph ¶
Contract 16: Connect UI elements

Deleted: the functionality

Deleted: the graph ¶

2.3. Dependencies

The following section describes how the component dependencies will impact development.

2.3.1. IngestionFunctionality

IngestionFunctionality is dependent on EventConfiguration for the retrieval of log files from the root and team directories, TranscriberAdapter to get text file transcriptions of audio and video files, and EnforcementActionReport to get the location and description of the log files that failed validation.

2.3.2. VersionControl

VersionControl is dependent on EventConfiguration to retrieve the Lead IP address and the number of connections to the Lead IP, and to identify the type of user in the system, and to connect to the network. Version control also depends on VectorDatabase to get the current version of the vectors, and Vector to get the local version of the vectors.

2.3.3. TableManager

TableManager is dependent on LogEntry to retrieve the log entry data that will be displayed in the log entry and vector table. It is also dependent on FilterFunctionality to display the filtered entries in the log entry table.

2.3.4. GraphFunctionality

GraphFunctionality is dependent on QTGraph to get specific UI elements for the node graph.

3. Detailed Descriptions

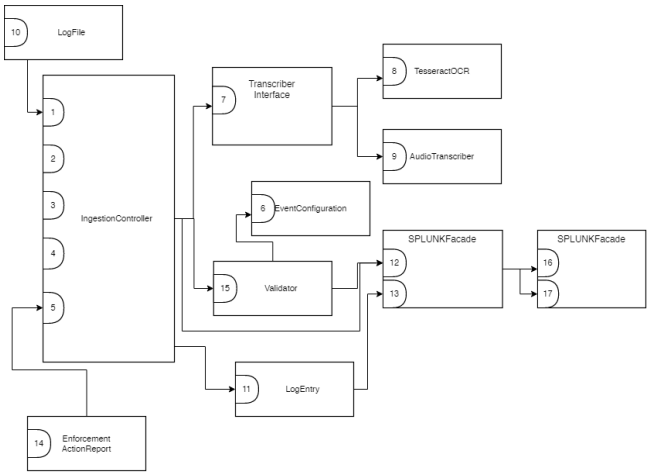
This section provides a detailed description of the identified subsystems of the PICK Tool, including their classes and contracts.

3.1. Ingestion Subsystem

This subsystem encloses all the actions taken by the PICK Tool in order to gather the logs from the root directory, cleanse, validate, upload to SPLUNK, and signal any errors pertaining to the log files. Due to these responsibilities, the subsystem includes the following classes:

1. IngestionController
2. EventConfiguration
3. TranscriberInterface
4. Log File
5. LogEntry
6. SPLUNKFaçade
7. EnforcementActionReport
8. Validator
9. SPLUNK

Diagram 2: UML System Diagram of the Ingestion Subsystem



3.1.1. Class Descriptions for IngestionController

3.1.1.1. IngestionController

Class Name: IngestionController
Superclass: N/A
Subclasses: N/A
Private Responsibilities • Control the complete ingestion process

• Schedule cleansing, validation and uploading of log files to SPLUNK • Sets properties for directory once structural check operation is done	
Contract: 1. Gather files from directories	
Responsibilities	**Collaborations**
1. Retrieve files from each directory	EventConfig(5)
Contract: 2. Provide ingestion status of log files	
Responsibilities	**Collaborations**
1. Know the ingestion status of every file	LogFile(8)
Contract: 3. Transcribe non-text files to text	
Responsibilities	**Collaborations**
1. Retrieve text files from audio/video files	TranscriberInterface
Contract: 4. Cleanse Files	
Responsibilities	**Collaborations**
1. Retrieve cleansed files from the text files	Cleanser
Contract: 5. Validate Files	
Responsibilities	**Collaborations**
1. Retrieve validated files 2. Retrieve non-validated files	Validator

3.1.1.1.1. Contract #1: Gather files from Directories

Contract Description: Gathers files from the directories and instantiates an object of LogFile for each file.

1. Get Files

-Signature:

- **Method name:**
  - **Python Syntax**: get_files()
  - **Complete:** public LogFile[] get_files()
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    N/A
  - **Output:**
    LogFile[]: List of LogFiles objects containing the filename, path, and a null status for: transcribed, cleansed, and validated.

Purpose: Create a list of files, with file status for each process in the ingestion process.

Collaborators: EventConfig, LogFile

Pre-condition:

- The event has been fully initialized
- The analyst has established connection with the host database

Post-condition:

- All files in the directories provided by the analyst are in a list of LogFile

3.1.1.1.2. Contract #2: Provide ingestion status of log files

Contract Description: Provides the ingestion status of each LogFile after each step in the ingestion process

Protocols:

1. Provide Ingestion Status

-Signature:

- **Method name:**
  - **Python Syntax**: get_status()
  - **Complete:** LogFile.get_status()

- **Description and type of input, output, input-output parameters:**
  - **Input:**
    The input will call the object of LogFile and call a mothod on itself to retrieve information.
  - **Output:**
    NA

Purpose: Provide the ingestion status of all files.
Collaborators: LogFile, TableManager
Pre-condition:
- The list of LogFile is initialized

Post-condition:
- N/A

3.1.1.1.3. Contract #3: Transcribe non-text files to text

Contract Description: Provides the ingestion status of each LogFile after each step in the ingestion process
Protocols:

1. Provide Ingestion Status

-Signature:
- **Method name:**
  - **Python Syntax**: get_status()
  - **Complete:** LogFile.get_status()
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    The input will call the object of LogFile and call a mothod on itself to retrieve information.
  - **Output:**
    NA

Purpose: Provide the ingestion status of all files.
Collaborators: LogFile, TranscriberInterface
Pre-condition:
- The list of LogFile is initialized

Post-condition:
- N/A

3.1.1.1.4. Contract #4: Cleanse Files

Contract Description: Cleanses the LogFile.
Protocols:

1. Cleanse Log File

-Signature:
- **Method name:**
  - **Python Syntax**: cleanse_files(LogFile)
  - **Complete:** public LogFile cleanse_files(LogFile)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    The input will send a LogFile object to be cleansed.
  - **Output:**
    A cleansed LogFile

Purpose: Cleanses the log files.
Collaborators: LogFile, Cleanser
Pre-condition:
- The list of LogFile is initialized

Post-condition:
- All LogFiles are cleansed.

3.1.1.1.5. Contract #5: Validate Files

Contract Description: Validates the LogFile and sends them to the appropriate place based on their validation status.

Protocols:

1. Validate Log File

-Signature:

- **Method name:**
  - **Python Syntax**: validate_files(LogFile)
  - **Complete:** public void validate_files(LogFile)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    The input will send a LogFile object to be cleansed.
  - **Output:**
    A cleansed LogFile

Purpose: Cleanses the log files.

Collaborators: LogFile, Validator, EnforcementActionReport, SPLUNKFacade

Pre-condition:

- The LogFile is cleansed.

Post-condition:

- The LogFile has a valid or invalid status.
- The invalid files are send to the Enforcement Action Report.

2. Send invalid files to the EnforcementActionReport

-Signature:

- **Method name:**
  - **Python Syntax**: send_to_report(InvalidFile)
  - **Complete:** public void send_to_reort(InvalidFile)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    The input will send list(,list of line numbers and error_messages)
  - **Output:**

    N/A

Purpose: Separate invalid from the list of LogFiles to be ingested

Collaborators: EnforcementActionReport

Pre-condition:

- The LogFile is validated.

Post-condition:

- The invalid files are sent to the Enforcement Action Report.

3. Send valid files to SPLUNKfacade

-Signature:

- **Method name:**
  - **Python Syntax**: send_to_report(InvalidFile)
  - **Complete:** public void send_to_reort(InvalidFile)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    The input will send list of (LogFile, and a list of line numbers and error_messages)
  - **Output:**

. **Purpose:** Gathers the log entries in the SPLUNK index into the LogEntry data type.

Collaborators: LogFile, EventConfig

Pre-condition:

- The LogFile is validated.

Post-condition:

- The invalid files are send to the Enforcement Action Report.

3.1.1.2. Class Description EventConfiguration

Class Name: EventConfiguration	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • Knows lead IP • Knows number of connections established • Knows if the user is a lead/analyst	
Contract: 6. Provide event data	
1. Knows event name, event description, event start date/time, event end date/time 2. Knows directories for the following folders: root, white, blue, red.	**N/A**

Contract #6: Provide Event Data

Contract Description: Provides the start and end time of the event

Protocols:

1. Get Start-End Time

-Signature:

- **Method name:**
  - **Python Syntax**: get_start_end_time()
- **Complete:** public List<Time> get_start_end_time()
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    N/A
  - **Output:**
    List containing the start and end time as Time objects.

Purpose: Get the time range of the event.
Collaborators: N/A
Pre-condition:
- The event has been initialized

Post-condition:
- A list containing the start and end time is returned.

2. Get Root Directory

-Signature:

- **Method name:**
  - **Python Syntax**: get_root_directory()
- **Complete:** public String get_root_directory()
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    N/A
  - **Output:**
    String containing the path of the root folder.

Purpose: Get the path to the root directory where the log files are contained.
Collaborators: N/A
Pre-condition:
- The event has been initialized
- The actual directory exists.

Post-condition:
- A string containing the path of the root folder is returned.

3.1.1.3. Class Description TranscriberInterface

Class Name: TranscriberInterface	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • Know formatting parameters for the OCR • Know formatting parameters for the audio transcriber	
Contract: 7. Provide transcribed log files	
Responsibilities	**Collaborations**
1. Transcribe audio/video log files 2. Transcribe image log files	AudioTranscriber TesseractOCR LogFile(8)

3.1.1.3.1. Contract #7: Provide Transcribed Log Files

Contract Description: Utilizes the transcribers to transcribe log files that are not directly readable.

Protocols:

1. Transcribe Log Files

-Signature:

- **Method name:**
  - **Python Syntax**: transcribe_log_file(filepath)
  - **Complete:** public String transcribe_log_file(String filepath)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    String containing the directory path of the file to be transcribed.
  - **Output:**
    String containing the text contents from the log file.

Purpose: Gather the contents of the audio, image and video log files into a readable format for the system.

Collaborators: LogFile

Pre-condition:

- The actual log file exists.

Post-condition:

- The content from the log file is transcribed into a string object.

3.1.1.4. Class Description TesseractOCR

Class Name: TesseractOCR	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • N/A	
Contract: 8. Transcribe Image Log Files	
Responsibilities	**Collaborations**
3. Transcribe image log files	N/A

3.1.1.4.1. Contract #8: Transcribe Image Log Files

Contract Description: Converts image log files into plain text.

Protocols:

1. Convert Image Log Files

-Signature:

- **Method name:**
  - **Python Syntax**: transcribe(filepath)
  - **Complete:** public String transcribe(String filepath)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    String containing the directory path of the file to be transcribed.
  - **Output:**
    String containing the text contents from the image file.

Purpose: Gather the contents of the image log files into a readable format for the system.
Collaborators: N\A
Pre-condition:
- The actual log file exists.

Post-condition:
- The content from the log file is transcribed into a string object.

3.1.1.5. Class Description Audio Transcriber

Class Name: Audio Transcriber	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities: N\A	
Contract: 9. Transcribe Audio Log Files	
Responsibilities	**Collaborations**
4. Transcribe audio files.	N/A

3.1.1.5.1. Contract #9: Transcribe Audio Log Files

Contract Description: Converts audio log files into plain text.

Protocols:

1. Convert Audio Log Files

-Signature:
- **Method name:**
  - **Python Syntax**: transcribe (filepath)
  - **Complete:** public String transcribe(String filepath)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    String containing the directory path of the file to be transcribed.
  - **Output:**
    String containing the text contents from the audio file.

Purpose: Gather the contents of the audio log files into a readable format for the system.
Collaborators: LogFile
Pre-condition:
- The actual log file exists.

Post-condition:
- The content from the log file is transcribed into a string object.

3.1.1.6. Class Description LogFile

Class Name: LogFile
Superclass: N/A
Subclasses: N/A
Private Responsibilities • N/A
Contract: 8. Provide data of an actual log file

Responsibilities	Collaborations
1. Know log file sourcetype 2. Know log file source 3. Know log file content 4. Know log file data type 5. Know log file path	IngestionController(4)
Contract: 10. Provide ingestion status of log file	
Responsibilities	**Collaborations**
1. Know ingestion status of log file	Validator(14)

3.1.1.6.1. Contract #10: Provide data of a Log File

Contract Description: Gets the log entries from the SPLUNK index pertaining to the event.

Protocols:

1. Get File Path

-Signature:

- **Method name:**
  - **Python Syntax**: get_file_path()
  - **Complete:** public String get_file_path()
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    N/A
  - **Output:**
    A string containing the file path of an actual log file in the directory.

Purpose: Provide the directory path of a log file.

Collaborators: N/A

Pre-condition:

- The actual log file exists.
- The directory is reachable from the system.

Post-condition:

- The returned string contains the full system directory path pertaining to the log file.

3.1.1.7. Class Description LogEntry

Class Name: LogEntry	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • N/A	
Contract: 11. Provide data of an occurrence	
Responsibilities	**Collaborations**
2. Know log entry ID 3. Know timestamp 4. Know source 5. Know sourcetype 6. Know host 7. Know content description	

3.1.1.8. Class Description SPLUNKFacade

Class Name: SPLUNKFacade
Superclass: N/A
Subclasses: N/A
Private Responsibilities

• N/A	
Contract: 12. Send log files to SPLUNK	
Responsibilities	**Collaborations**
1. Upload log files to be parsed and stored in SPLUNK	SPLUNK(15)
Contract: 13. Provide SPLUNK log entries	
Responsibilities	**Collaborations**
1. Forward the SPLUNK log entries	SPLUNK(15)

3.1.1.8.1. Contract #12: Send Log Files to SPLUNK

Contract Description: Sends the log files to a specific index in SPLUNK.

Protocols:

1. Import Log File to Splunk

-Signature:

- **Method name:**
  - **Python Syntax**: import_log_file_to_splunk(log_file)
  - **Complete:** public void import_log_file_to_splunk(LogFile log_file)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    The input will send a LogFile object pertaining to the file to be ingested into SPLUNK.
  - **Output:**
    N/A

Purpose: Import the files in SPLUNK for parsing and storage.

Collaborators: LogFile, EventConfig

Pre-condition:

- The log file has been cleansed.
- The log file has been validated or the user has purposely skipped validation of it.
- The event has been initialized.
- The index pertaining to the event has been initialized.
- The SPLUNK service has been initialized.

Post-condition:

- The log entries inside the log file has been parsed and stored in the SPLUNK index.

3.1.1.8.2. Contract #13: Provide Log Entries from SPLUNK

Contract Description: Gets the log entries from the SPLUNK index pertaining to the event.

Protocols:

1. Get SPLUNK Log Entries

-Signature:

- **Method name:**
  - **Python Syntax**: get_splunk_log_entries()
  - **Complete:** public List<LogEntry> get_splunk_log_entries()
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    N/A
  - **Output:**
    A list of type LogEntry containing all the entries stored in the SPLUNK index of the event.

Purpose: Gather the log entries from the SPLUNK index of the event.

Collaborators: EventConfig

Pre-condition:

- The event has been initialized
- The SPLUNK index of the event has been initialized

- The SPLUNK service has been initialized.

Post-condition:

- The log entries in the SPLUNK index of the event are gathered in a list of type LogEntry.

3.1.1.9. Class Description EnforcementActionReport

Class Name: EnforcementActionReport	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • N/A	
Contract: 14. Provide the list of the errors during the ingestion	
Responsibilities	**Collaborations**
1. Knows the ingestion error's line 2. Knows the ingestion error's description	SPLUNKFacade(10) IngestionFunctionality(2) LogFile(9)

3.1.1.9.1. Contract #14: Provide Ingestion Error List

Contract Description: Provides the errors occurred during the ingestion process of each LogFile

Protocols:

1. Get Ingestion Errors

-Signature:

- **Method name:**
  - **Python Syntax**: get_ingestion_errors()
  - **Complete:** public List<List<Dynamic>> get_ingestion_errors()
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    N/A
  - **Output:**
    A 2d list of Strings containing the line where the error occurred and the error message per each log file.

Purpose: Document the errors that happened during the ingestion process

Collaborators: Validator, Cleanser

Pre-condition:

- The event has been initialized
- Log files have been submitted to the ingestion process

Post-condition:

- The 2d list contains the errors is current to the status of the system.

3.1.1.10. Class Description Validator

Class Name: Validator	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • Knows interaction parameters for validation script calling • Knows the timestamp formats	
Contract: 13. Send invalidinvalidinvalid files to EnforcementActionReport	
Responsibilities	**Collaborations**
1. Send Invalid filesfilesfiles	EnforcementActionReportEnforcementActionReportEnforcementActionReport

Contract: 15. Send valid files to SPLUNKInterfacerSPLUNKInterfacerSPLUNKInterfacer	
Responsibilities	**Collaborations**
1. Knows the valid filesfilesfiles	SPLUNKInterfacerSPLUNKInterfacerSPLUNKInterfacer

3.1.1.10.1. Contract #15: Send invalid files to the EnforcementActionReport

Contract Description: Sends invalid files to the enforcement action report.
Protocols:
1. Send Files
-Signature:
- **Method name:**
  - **Python Syntax**: send_invalid_files(LogFile, line_errors)
  - **Complete:** public void send_invalid_files(LogFile, line_errors)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    Name of the log file, and a list of lists with 2 elements: line_number and error_message.
  - **Output:**
    N/A

Purpose: Send the report a list of invalid files
Collaborators: LogFile, EnforcementActionReport
Pre-condition:
- The event has been fully initialized
- The analyst has established connection with the host database
- The LogFile has gone through the validation process with an invalid status

Post-condition:
- The invalid files are sent to the EnforcementActionReport.

3.1.1.10.2. Contract #: Send valid files to SPLUNKfacade

Contract Description: Sends valid files to SPLUNKfacade to be ingested.
Protocols:
1. Send Files
-Signature:
- **Method name:**
  - **Python Syntax**: send_valid_files(LogFile)
  - **Complete:** public void send_valid_files(LogFile)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    Valid LogFile
  - **Output:**
    N/A

Purpose: Sends SPLUNK the validated LogFiles
Collaborators: LogFile, SPLUNKFacade
Pre-condition:
- The event has been fully initialized
- The analyst has established connection with the host database
- The LogFile has gone through the validation process with a valid status

Post-condition:
- The valid files are sent to SPLUNKfacade.

3.1.1.11. Class Description SPLUNK

Class Name: SPLUNK	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • Checks for deltas in directories • Filters through log entries • Searches through log entries	
Contract: 16. Convert log files intointo log entries	
Responsibilities	**Collaborations**
1. Parse log files 2. Store log entries	
Contract: 17. Query SPLUNK Index	
Responsibilities	**Collaborations**
3. Filter log entries on the index 4. Export log entries from the index	

3.1.1.11.1. Contract #N: Convert Log Files into Log Entries

Contract Description: Parse the log files into log entries and store them in the

Protocols:

1. Import Log File to Splunk

-Signature:

- **Method name:**
  - **Python Syntax**: upload(filepath)
  - **Complete:** public void upload(String filepath)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    A string containing the full system path of the log file to be ingested.
  - **Output:**
    N/A

Purpose: Import the files in SPLUNK for parsing and storage.

Collaborators: LogFile, EventConfig

Pre-condition:

- The index has been initialized.
- The SPLUNK service has been initialized.

Post-condition:

- The log entries inside the log file has been parsed and stored in the SPLUNK index.

Contract #N: Query SPLUNK Index

Contract Description: Gets the log entries that satisfy the input arguments of the query from an index.

Protocols:

1. Export Entries

-Signature:

- **Method name:**
  - **Python Syntax**: export(kwargs)
  - **Complete:** public Results transcribe(Dict<String> kwargs)
- **Description and type of input, output, input-output parameters:**
  - **Input:**

Dictionary of strings containing the parameters for the SPLUNk query.

- **Output:**

A Results type object containing all of the gathered log entries from the query.

Purpose: Gather the log entries from a SPLUNK index pertaining to a filter query.
Collaborators: N\A
Pre-condition:

- The SPLUNK index has been initialized.
- The SPLUNK service is running.

Post-condition:

- The Results object contains all of the log entries that satisfy the filter query.

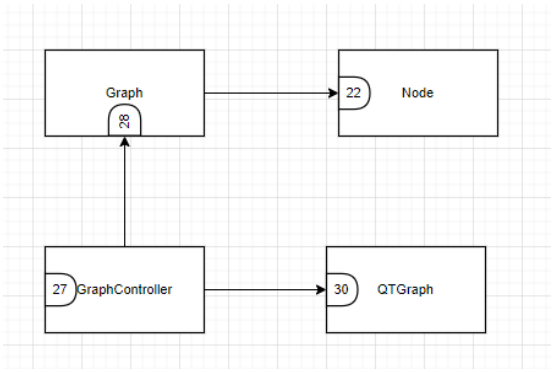
3.2. Graph Subsystem

The graph subsystem updates the information of the nodes displayed in the graph of the vector and controls the elements of graph UI. Due to these responsibilities the subsystem includes the following classes:

1. GraphFunctionality
2. QTGraph
3. Graph
4. Node

Deleted: ¶
<#>Scenario 2: Correlating Activity to Response¶
¶
Deleted: <#>Scenario 3: Generate Graph¶

Diagram 3: UML Graphing Sub-System Component Diagram



3.2.1. Class Descriptions for Graph Subsystem

3.2.1.1. Class description GraphFunctionality

Class Name: GraphFunctionality
Superclass: N/A
Subclasses: N/A
Private Responsibilities • Manage node information

Contract: 16. Provide functionality for editing graph data	
Responsibilities	**Collaborations**
1. Manage node information	Graph QTGraph

3.2.1.1.1. Contract G1: Provide functionality for editing graph data

Contract Description: Connects UI inputs to make edits to the data that makes up the graph

Protocols:

1. Manage Graph

-Signature:

- **Method name:**
  - **Python Syntax**: manage_graph(Graph, changes)
  - **Complete:** public void manage_graph(Graph, changes)
- **Description and type of input, output, input-output parameters:**
  - **Input:**

    Name of the vector/graph to be edited, and a list of changes to be applied to the graph data
  - **Output:**

    N/A

Purpose: Apply user changes to the data that makes up a graph

Collaborators: Graph, QTGraph

Pre-condition:

- Valid graph name

Post-condition:

- N/A

3.2.1.2. Class description QTGraph

Class Name: QTGraph	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • Interact with QTGraph API to visually represent graph data	
Contract: 17. Interact with QTGraph API to create a graphical representation of nodes and relationships	
Responsibilities	**Collaborations**
1. Connect modified data to QTGraph functionality	GraphFunctionality

3.2.1.2.1. Contract G2: Interact with QTGraph API to create a graphical representation of nodes and relationships

Contract Description: Interact with QTGraph API to connect changes made to data and visually represent graph data

Protocols:

1. QTGraphFunctionality

-Signature:

- **Method name:**
  - **Python Syntax**: qt_functionality(function, params)
  - **Complete:** public QTGraph qt_functionality(function, params)
- **Description and type of input, output, input-output parameters:**
  - **Input:**

Name of the function to be used, and a list of any parameters required for it.

  - **Output:**

QTGraph item

Deleted: #: Send invalid files to the EnforcementActionReport

Deleted: Sends invalid files to the enforcement action report.

Deleted: Send Files

Deleted: send_invalid_files(LogFile, line_errors

Deleted: send_invalid_files(LogFile, line_errors

Deleted: log file

Deleted: lists with 2 elements: line_number and error_message.

Deleted: Send the report

Deleted: list of invalid files

Deleted: LogFile, EnforcementActionReport

Deleted: <#>The LogFile has gone through the validation process with an invalid status¶

Deleted: <#>The invalid files are sent to the EnforcementActionReport.¶

Deleted: Interacts

Purpose: Connect UI inputs to functionality of QTGraph API
Collaborators: GraphFuncionality
Pre-condition:
- Valid QTGraph function and necessary parameters.

Post-condition:
- N/A

3.2.1.3. Class description Graph

Class Name: Graph	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • Store position information for nodes • Store position information for relationships of nodes • Store graph display/export information • Store information about graph nodes	
Contract: 18. Store position information needed for graphical representation	
Responsibilities	**Collaborations**
1. Store position information for nodes 2. Store position information for relationships of nodes 3. Store graph display/export information	GraphFunctionality
Contract: 19. Store information about the nodes that belong to the graph	
Responsibilities	**Collaborations**
1. Store information about graph nodes	Node

3.2.1.3.1. Contract G3: Store position information needed for graphical representation

Contract Description: Connects UI inputs to make edits to the data that makes up the graph
Protocols:
1. Store Graph position
-Signature:
- **Method name:**
  - **Python Syntax**: store_graph_position(Graph, position, type)
  - **Complete:** public void store_graph_position(Graph, position, type)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    Name of the vector/graph being edited, position to be saved, and the type of graphical element it is.
  - **Output:**
    N/A

Purpose: Apply user changes to the data that makes up a graph
Collaborators: GraphFunctionality
Pre-condition:
- Valid graph name, coordinates in the correct format

Post-condition:
- N/A

3.2.1.3.2. Contract G4: Store information about the nodes that belong to the graph

Contract Description: Store information of which nodes belong to specific graph
Protocols:
1. Update Graph
-Signature:
- **Method name:**
  - **Python Syntax**: update_graph(Node, changes)
  - **Complete:** public void update_graph(Node, changes)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    Name of the node, and a list of changes to be applied to the graph data
  - **Output:**
    N/A

Purpose: Apply user changes to the nodes in the graph
Collaborators: Node
Pre-condition:
- Valid node name

Post-condition:
- N/A

3.2.1.4. Class description Node

Class Name: Node	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • Represents a relevant part of a vector event • Stores information relevant to the origin of the action (red, white, blue)	
Contract: 18. Store information for individual nodes	
Responsibilities	**Collaborations**
1. Represents a relevant part of a vector event 2. Stores information relevant to the origin of the action (red, white, blue)	Graph

3.2.1.4.1. Contract G5: Store information for individual nodes

Contract Description: Stores information for specific nodes
Protocols:
1. Update Node
-Signature:
- **Method name:**
  - **Python Syntax**: update_node(changes)
  - **Complete:** public void update_node(changes)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    list of changes to be applied to the graph node
  - **Output:**
    N/A

Purpose: Apply user changes to the node
Collaborators: Graph
Pre-condition:
- Valid node name

Post-condition:
- N/A

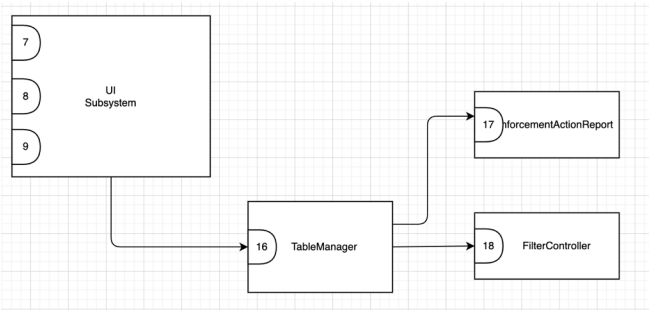
Deleted: about
Deleted: **Contract:** 18. Store position information needed for graphical representation ... [40]
Deleted: position
Deleted: ¶
<#>Store position information for relationships of nodes¶
Store graph display/export information
Deleted: GraphFunctionality
Deleted: **Contract:** 19. Store information about the nodes that belong to the graph ... [41]
Deleted: Store
Deleted: about graph nodes
Deleted: ¶

3.3. UI Subsystem

This subsystem is the most fluent when it comes to sustaining the system's functionality. Included are the classes of TableManager, Vector, UIFunctionality, LogEntry, FilterFunctionality and UIView. In this case, the primary focus is the TableManager.

Contract 9: Get log entry data
Contract 10: Display log entry data in the vector table
Contract 11: Get filtered entries

Diagram 4: UML Table User Interface Sub-System Component Diagram



3.3.1. Class Descriptions for UI Subsystem

Deleted: ¶
¶

3.3.1.1. Table Manager

Class Name: TableManager	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • Allows user to add, delete, modify log entries • Allows user to search and filter through log entries	
Contract: 9 Get log entry data	
Responsibilities	**Collaborations**
3. Represents current log entry data 4. Table must update with given set of user actions of add, delete, edit	Log File(8)

3.3.1.1.1. Contract 1: Provide log entry data on table

Contract Description: Uses current stored data to list all log entries associated with the vector
Protocols:
1. ManageTable
-Signature:
- **Method name:**
  - **Python Syntax**: table_enties(Vector, logEntries)
  - **Complete:** public void table_entries(Vector, logEntries)
- **Description and type of input, output, input-output parameters:**
  - **Input:**
    - Vector Name
  - **Output:**

- Log Entries

Purpose: UI of table with appropriate log entries

Collaborators: LogEntry, Vector

Pre-condition:

- Vector Name

Post-condition:

N/A

Deleted: ¶

3.3.1.2. UIView

Class Name: UIFunctionality
Superclass: N/A
Subclasses: N/A
Responsibilities
1. Displays view of each UI specified by the user
2. Allows user to search and filter through log entries
3. Displays data
4. Allows user to add, delete, modify log entries
Collaborators
GraphInterface
VectorViewInterface
ConfigConfiguration
LogViewEnforcementActionReportLog File(8)
VectorViewLog Entry

Deleted: ¶
<#>UIFunctionality¶

Deleted: UIView

Deleted: **Superclass**: N/A ... [42]

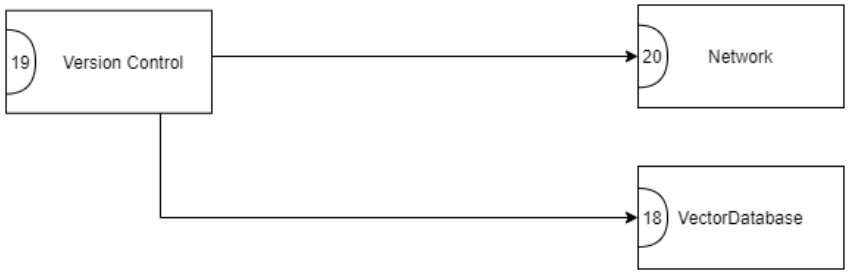
Deleted: Knows the occurrence¶
Knows the associated log entries

3.4. Network Subsystem

The network subsystem establishes the connection between the analyst DB and the analyst DB, check for deltas in the host DB and updates the information in the vectors. Due to these responsibilities, the subsystem includes the following classes:

1. NetworkFunctionality
2. VersionControl
3. VectorDatabase

Diagram 5: UML_Network Sub-System Component Diagram



3.4.1. Class Descriptions for Network Subsystem

Deleted: ¶

Deleted: SubsystemQ

3.4.1.1. Class Description NetworkFunctionality

Class Name: NetworkFunctionality
Superclass: N/A
Subclasses: N/A

Private Responsibilities • N/A	
Contract: 20. Establish connection to DB	
Responsibilities	**Collaborations**
2. Connect to analyst and host DB 3. Send analyst commits to lead 4. Send updated vectors to analyst	EventConfiguration(5)

3.4.1.1.1. Contract #20: Establish Connection to DB

Contract Description: Connect analyst local DB to lead DB

Protocols:

4. Establish Connection

-Signature:

- **Method name:**
  - **Python Syntax:** connectToLeadDB()
  - **Complete:** EventConfiguration.connectToLeadDB(vectorDBAnalyst, vectorDBLead)
- **Description and type of input, output, input-output parameters:**
  - **Input:** Analyst IP address, Lead Analyst IP address
  - **Output:** N/A

Purpose: Establish connection to lead DB

Collaborators: EventConfiguration

Pre-condition:

- The local DB must be an analyst; it cannot be the lead
- The IP address must be different from the lead

Post-condition:

- The local DB is connected to the lead DB

3.4.1.2. Class Description VersionControl

Class Name: VersionControl	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • N/A	
Contract: 19. Store DB Data	
Responsibilities	**Collaborations**
5. Knows and tracks changes in the vector of the host DB 6. Knows changes that need approval from lead analyst 7. Knows different versions of the vector	

3.4.1.2.1. Contract #19: Store DB Data

Contract Description: Check for changes in vectors in the host DB and stores them into the host DB

Protocols:

1. Provide DB data vectors

-Signature:

- **Method name:**
  - **Python Syntax:** checkDeltas()
  - **Complete:** VersionControl.checkDeltas()

- **Description and type of input, output, input-output parameters:**
  - **Input:**
  - **Output:**

Purpose: Find changes in the vectors of the hostDB and update them in the hostDB
Collaborators: N/A
Pre-condition:
- N/A

Post-condition:
- Vectors in the host DB are updated if changes are found; vectors remain the same otherwise.

3.4.1.3. Class Description VectorDatabase

Class Name: VectorDatabase	
Superclass: N/A	
Subclasses: N/A	
Private Responsibilities • N/A	
Contract: 18. Store vectors of event	
Responsibilities	**Collaborations**
5. Knows the vectors of the event	Vector (23)

3.4.1.3.1. Contract #18: Store Vectors of Event

Contract Description: Get vectors from event and stores them in the vector database.
Protocols:
1. Provide vectors from event
-Signature:
- **Method name:**
  - **Python Syntax:** getEventVector(Event)
  - **Complete:** Vector.getEventVector(Event)
- **Description and type of input, output, input-output parameters:**
  - **Input:** Event
  - **Output:** Vector from Event

Purpose: Gets vector from specific event
Collaborators: Vector
Pre-condition:
- Event must have a vector populated with nodes

Post-condition:
- Vector from specified event is stored in the Vector Database

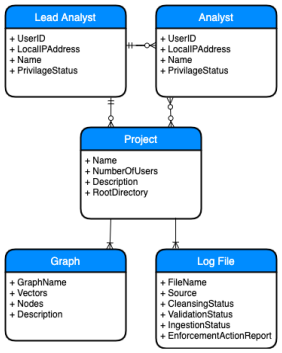
4. Database

This section describes the database aspect of the system used to associate with input and output data. Classes from each of the entities are carefully evaluated to determine whether itis requires database access to manipulate data. Included are the Database Schema and the determinable Entity-Relational Model.

4.1. Database Schema

The Database Schema presents the blueprint as to how data is to be collected and shared among classes in the database. This organization allows for the applications of tables and views to be applied in each of the classes.

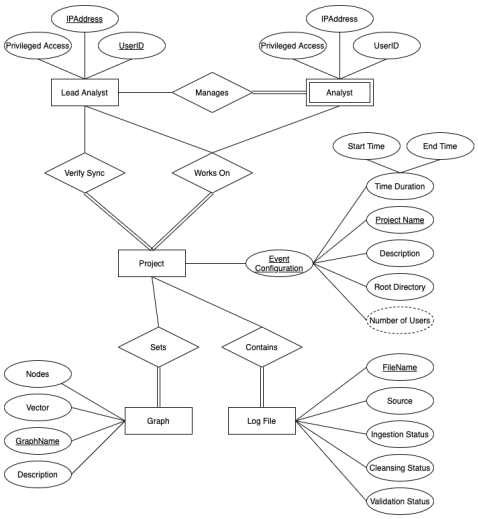
Diagram 3: Database Schema



4.2. Entity-Relational Model

The Entity-Relational Model displays the entities in which the system is associated with on a database scale. Each entity presents its respectable attributes and relationships among other entities. This model helps with determining why data among each entity is in relationship with each other, with regards to the type of relationship shared.

Diagram 4: UML Entity-Relational Model


IPAddress
Privileged Access
UserID
Lead Analyst
Manages
IPAddress
Privileged Access
UserID
Analyst
Verify Sync
Works On
Start Time
End Time
Time Duration
Project Name
Description
Root Directory
Number of Users
Project
Event Configuration
Sets
Contains
Nodes
Vector
GraphName
Description
Graph
Log File
FileName
Source
Ingestion Status
Cleansing Status
Validation Status

*

Page iv: [1] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [2] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [3] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [4] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [5] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [6] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [7] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [8] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [9] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [10] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [11] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [12] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [13] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [14] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [15] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [16] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [17] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [18] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [19] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [20] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [21] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [22] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [23] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [24] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [25] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [26] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [27] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [28] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [29] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [30] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [31] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [32] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [33] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM
Page iv: [34] Deleted	Iglesias, Matthew	3/31/20 8:45:00 AM

Page iv: [35] Deleted	**Iglesias, Matthew**	**3/31/20 8:45:00 AM**
Page iv: [36] Deleted	**Iglesias, Matthew**	**3/31/20 8:45:00 AM**
Page iv: [37] Deleted	**Iglesias, Matthew**	**3/31/20 8:45:00 AM**
Page iv: [38] Deleted	**Iglesias, Matthew**	**3/31/20 8:45:00 AM**
Page iv: [39] Deleted	**Iglesias, Matthew**	**3/31/20 8:45:00 AM**
Page 22: [40] Deleted	**Iglesias, Matthew**	**3/31/20 8:58:00 AM**
Page 22: [41] Deleted	**Iglesias, Matthew**	**3/31/20 8:58:00 AM**
Page 24: [42] Deleted	**Rincon, Diego**	**3/31/20 9:34:00 AM**