

IBM Surveillance Insight for Financial
Services
Version 2.0.0

*IBM Surveillance Insight for Financial
Services Solution Guide*



Note

Before using this information and the product it supports, read the information in [“Notices” on page 125](#).

Product Information

This document applies to Version 2.0.0 and may also apply to subsequent releases.

Licensed Materials - Property of IBM

© **Copyright International Business Machines Corporation 2016, 2017.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Introduction..... V

Chapter 1. IBM Surveillance Insight for Financial Services..... 1
The solution architecture..... 1

Chapter 2. Surveillance Insight Workbench..... 5
Dashboard page..... 5
Alert Details page..... 6
Employee Details page..... 10
Notes page..... 10
Search pages..... 13

Chapter 3. Trade surveillance..... 17
Trade Toolkit data schemas..... 18
 Ticker price schema..... 18
 Execution schema..... 18
 Order schema..... 20
 Quote schema..... 21
 Trade schema..... 23
 End of day (EOD) schema..... 23
 Event schema..... 24
 Event data schema..... 24
 Audio metadata schema..... 25
Trade Surveillance Toolkit..... 25
Pump-and-dump use case..... 28
Spoofing detection use case..... 30
Extending Trade Surveillance..... 31

Chapter 4. E-Comm surveillance..... 35
E-Comm Surveillance Toolkit..... 35
E-Comm data ingestion..... 38
Data flow for e-comm processing..... 39
E-Comm use case..... 41
E-Comm Toolkit data schemas..... 44
 Party view..... 46
 Communication view..... 48
 Alert view..... 50
 Trade view..... 52
Extending E-Comm Surveillance..... 53

Chapter 5. Voice surveillance..... 55
Voice Surveillance metadata schema for the WAV Adaptor..... 57
WAV format processing..... 57
PCAP format processing..... 58

Chapter 6. NLP libraries..... 61
Emotion Detection library..... 61
Concept Mapper library..... 63
Classifier library..... 66

Chapter 7. Inference engine..... 71

Inference engine risk model.....	71
Run the inference engine.....	73
Chapter 8. Indexing and searching.....	75
Chapter 9. API reference.....	77
Alert service APIs.....	77
Notes service APIs.....	85
Party service APIs.....	90
CommServices APIs.....	93
Policy service APIs.....	94
Chapter 10. Develop your own use case.....	99
Chapter 11. Troubleshooting.....	103
Solution installer is unable to create the chef user.....	103
NoSuchAlgorithmException.....	103
java.lang.ClassNotFoundException: com.ibm.security.pkcsutil.PKCSException.....	104
java.lang.NoClassDefFoundError: com/ibm/security/pkcs7/Content.....	105
java.lang.NoClassDefFoundError: org/apache/spark/streaming/kafka010/LocationStrategies.....	105
java.lang.NoClassDefFoundError (com/ibm/si/security/util/SISecurityUtil).....	105
org.json.JSONException:A JSONObject text must begin with '{' at character 1.....	107
javax.servlet.ServletException:Could not find endpoint information.....	108
DB2INST1.COMM_POLICY is an undefined name.....	109
Authorization Error 401.....	109
javax.security.sasl.SaslException: GSS initiate failed.....	109
PacketFileSource_1_out0.so file.....:undefined symbol:libusb_open.....	110
[Servlet Error]-[SIFSRestService]: java.lang.IllegalArgumentException.....	110
Failed to update metadata after 60000 ms.....	110
java.lang.NoClassDefFoundError: com/sun/jersey/api/client/config/ClientConfig.....	110
DB2 SQL Error: SQLCODE=-204, SQLSTATE=42704, SQLERRMC=DB2INST1.COMM_TYPE_MASTER.....	111
org.apache.kafka.common.config.ConfigException: Invalid url in bootstrap.servers.....	111
[localhost/127.0.0.1, localhost/0:0:0:0:0:0:1] failed: Connection refused.....	111
java.lang.Exception: 500 response code received from REST service.....	112
javax.security.auth.login.FailedLoginException: Null.....	113
Caused by: java.lang.ClassNotFoundException: com.ibm.security.pkcs7.Content.....	114
No Audio files to play in UI – Voice.....	114
org.apache.kafka.common.KafkaException: Failed to construct kafka producer.....	114
CDISR5030E: An exception occurred during the execution of the PerfLoggerSink operator.....	115
CDISR5023E: The error is: No such file or directory.....	115
java.lang.Exception: 404 response code received from REST service!!.....	116
java.lang.NoClassDefFoundError: com/ibm/si/security/kafka/SISecureKafkaProducer.....	117
java.lang.NumberFormatException: For input string: "e16".....	118
CDISP0180E ERROR: The error is:The HADOOP_HOME environment variable is not set.....	119
Database connectivity issues.....	119
status":"500","message":"Expecting '{' on line 1, column 0 instead, obtained token: 'Token:.....	119
Runtime failures occurred in the following operators: FileSink_7.....	121
Subject and email content is empty.....	121
Appendix A. Accessibility features.....	123
Notices.....	125
Index.....	127

Introduction

Use IBM® Surveillance Insight for Financial Services to proactively detect, profile, and prioritize non-compliant behavior in financial organizations. The solution ingests unstructured and structured data, such as trade, electronic communication, and voice data, to flag risky behavior. Surveillance Insights helps you investigate sophisticated misconduct faster by prioritizing alerts and reducing false positives, and reduces the cost of misconduct.

Some of the key problems that financial firms face in terms of compliance misconduct include:

- Fraudsters using sophisticated techniques thereby making it hard to detect misconduct.
- Monitoring and profiling are hard to do proactively and efficiently with constantly changing regulatory compliance norms.
- A high rate of false positives increases the operational costs of alert management and investigations.
- Siloed solutions make fraud identification difficult and delayed.

IBM Surveillance Insight for Financial Services addresses these problems by:

- Leveraging key innovative technologies, such as behavior analysis and machine learning, to proactively identify abnormalities and potential misconduct without pre-defined rules.
- Using evidence-based reasoning that aids streamlined investigations.
- Using risk-based alerting that reduces false positives and negatives and improves the efficiency of investigations.
- Combining structured and unstructured data from different siloed systems into a single platform to perform analytics.

IBM Surveillance Insight for Financial Services takes a holistic approach to risk detection and reporting. It combines structured data such as stock market data (trade data) with unstructured data such as electronic emails and voice data, and it uses this data to perform behavior analysis and anomaly detection by using machine learning and natural language processing.

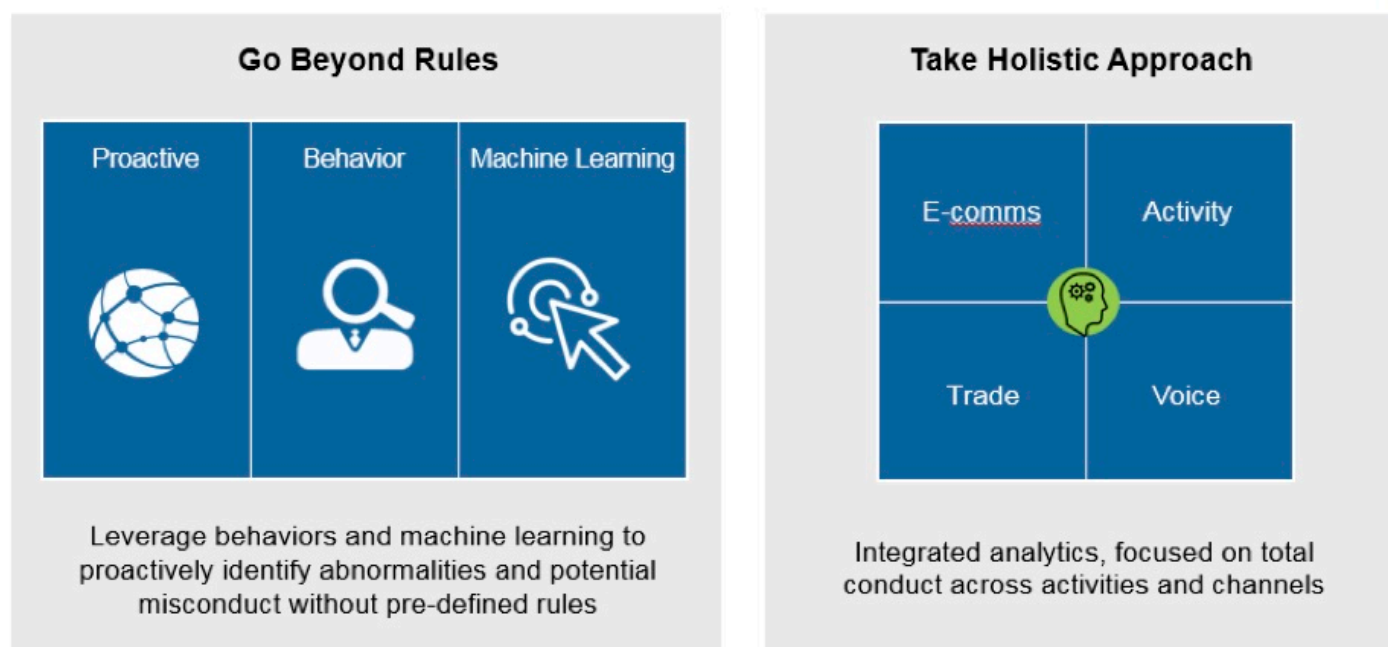


Figure 1: Surveillance Insight overview

Audience

This guide is intended for administrators and users of the IBM Surveillance Insight for Financial Services solution. It provides information on installation and configuration of the solution, and information about using the solution.

Finding information and getting help

To find product documentation on the web, access [IBM Knowledge Center](http://www.ibm.com/support/knowledgecenter) (www.ibm.com/support/knowledgecenter).

Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products. Some of the components included in the IBM Surveillance Insight for Financial Services have accessibility features. For more information, see [Appendix A, “Accessibility features,” on page 123](#).

The HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of features or functionality remain at the sole discretion of IBM.

Samples disclaimer

Sample files may contain fictional data manually or machine generated, factual data that is compiled from academic or public sources, or data that is used with permission of the copyright holder, for use as sample data to develop sample applications. Product names that are referenced may be the trademarks of their respective owners. Unauthorized duplication is prohibited.

Chapter 1. IBM Surveillance Insight for Financial Services

IBM Surveillance Insight for Financial Services provides you with the capabilities to meet regulatory obligations by proactively monitoring vast volumes of data for incriminating evidence of rogue trading or other wrong-doing through a cognitive and holistic solution for monitoring all trading-related activities. The solution improves current surveillance process results and delivers greater efficiency and accuracy to bring the power of cognitive analysis to the financial services industry.

The following diagram shows the high-level IBM Surveillance Insight for Financial Services process.



Figure 2: High-level process

1. As a first step in the process, data from electronic communications (such as email and chat), voice data, and structured stock market data are ingested into IBM Surveillance Insight for Financial Services for analysis.
2. The data is analyzed.
3. The results of the analysis are risk indicators with specific scores.
4. The evidences and their scores are used by the inference engine to generate a consolidated score. This score indicates whether an alert needs to be created for the current set of risk evidences. If needed, an alert is generated and associated with the related parties and stock market tickers.
5. The alerts and the related evidences that are collected as part of the analysis can be viewed in the IBM Surveillance Insight for Financial Services Workbench.

After the alerts are created and the evidences are collected, the remaining steps in the process are completed outside of IBM Surveillance Insight for Financial Services. For example, case investigators must work on the alerts and confirm or reject them, and then investigation reports must be sent out to the regulatory bodies as is required by compliance norms.

The solution architecture

IBM Surveillance Insight for Financial Services is a layered architecture made up of several components.

The following diagram shows the different layers that make up the product:

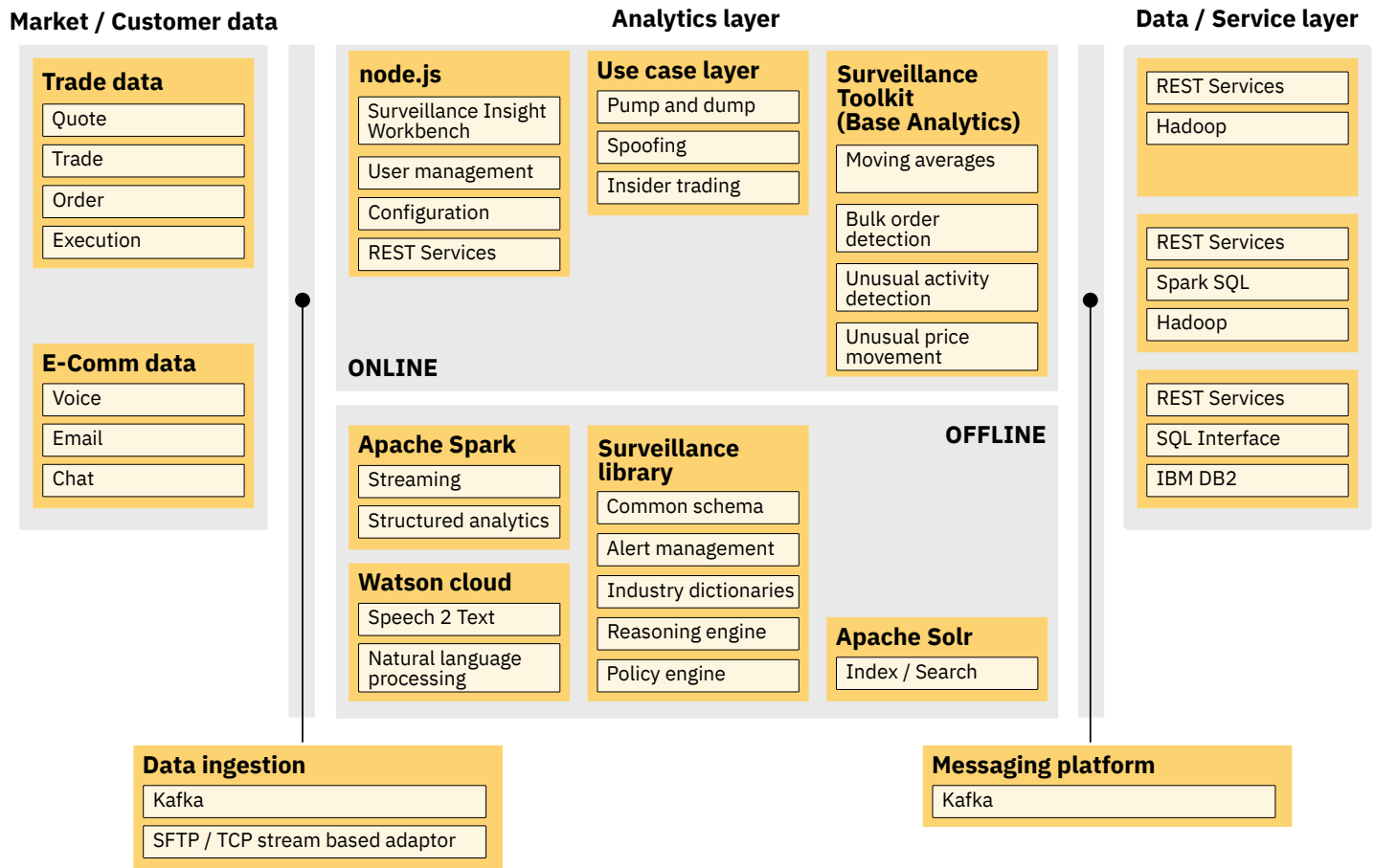


Figure 3: Product layers

- The data layer shows the various types of structured and unstructured data that is consumed by the product.
 - The data ingestion layer contains the FTP/TCP-based adaptor that is used to load data into Hadoop. The Kafka messaging system is used for loading e-communications into the system.
- Note:** IBM Surveillance Insight for Financial Services does not provide the adaptors with the product.
- The analytics layer contains the following components:
 - The Workbench components and the supporting REST services for the user interfaces.
 - Specific use case implementations that leverage the base toolkit operators.
 - The surveillance library that contains the common components that provide core platform capabilities such as alert management, reasoning, and the policy engine.
 - The Spark Streaming API is used by Spark jobs as part of the use case implementations.
 - Speech 2 Text and the NLP APIs are used in voice surveillance and eComms surveillance.
 - Solr is used to index content to enable search capabilities in the Workbench.
 - Kafka is used as an integration component in the use case implementations and to enable asynchronous communication between the Streams jobs and the Spark jobs.
 - The data layer primarily consists of data in Hadoop and IBM DB2®. The day-to-day market data is stored in Hadoop. It is accessed by using the spark-sql or spark-graphx APIs. Data in DB2 is accessed by using traditional relational SQL. REST Services are provided for data that needs to be accessed by the user interfaces and for certain operations such as alert management.

The following diagram shows the end-to-end component interactions in IBM Surveillance Insight for Financial Services.

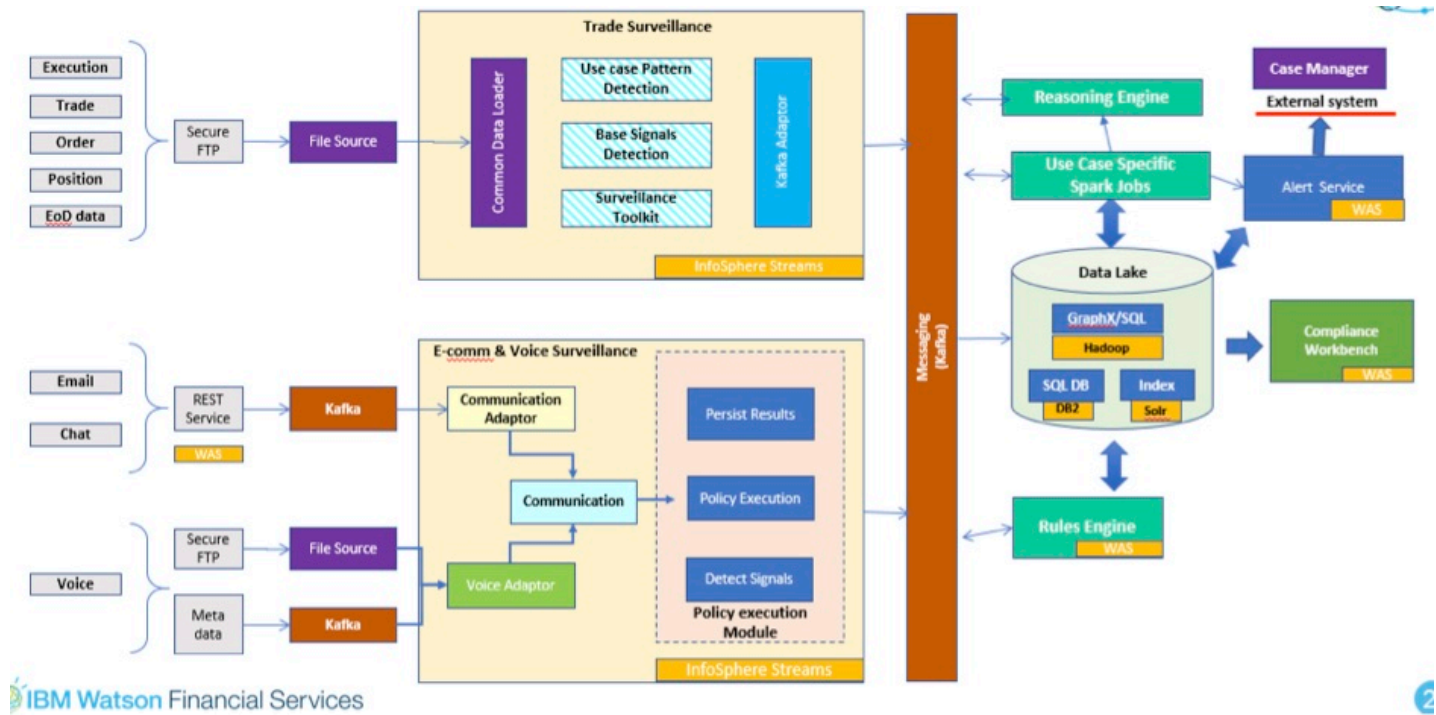


Figure 4: End-to-end component interaction

- Trade data is loaded into Hadoop through secure FTP. The Data Loader Streams job monitors specific folders in Hadoop and provides the data to the use cases that need market data.
- The trade use case implementations analyze the data and creates relevant risk evidences.
- Email and chat data is brought into the system through a REST service that drops the data from third-party sources into the Kafka topic.
- The unstructured data is analyzed by the Streams jobs and the results are persisted to Kafka.
- Voice data is obtained through secure FTP. The trigger for processing the data is then passed on through the Kafka message that contains the metadata about the voice data that needs to be processed.
- After the voice data is converted to text, the rest of the analysis is performed in the same way as the email and chat data is processed.
- The output, or the risk evidences from the use case implementations (trade, ecomm, and voice), are dropped into the Kafka messaging topics for the use case-specific Spark jobs. The Spark jobs perform the post processing after the evidences are received from the Streams jobs.

Chapter 2. Surveillance Insight Workbench

Users access the product through the Surveillance Insight Workbench, a web-based interface that provides users with the results of the analysis that is performed by the solution.

Dashboard page

The Dashboard page shows the Alerts and Employees tabs.

Alerts tab

The **Alert** tab shows open alerts that were created in the past 30 days from the date that the last alert was created. The results are sorted by risk score.

IBM Surveillance Insight 🔍 👤 BALAJI BALAJI ▾

ALERTS EMPLOYEES

Score	Type	Date	ID	Ticker	Asset Class	Status
100%	Spoofing	04/10/17	1490696753	KO	Equity	New
100%	Party Behavior Risk	04/14/17	1492169598173			New
100%	Party Behavior Risk	04/14/17	1492169825619			New
100%	Party Behavior Risk	04/14/17	1492169825890			New
100%	Party Behavior Risk	04/14/17	1492169861740			New
100%	Party Behavior Risk	04/14/17	1492169871292			New
100%	Party Behavior Risk	04/14/17	1492169873969			New
100%	Party Behavior Risk	04/14/17	1492169876118			New
100%	Party Behavior Risk	04/14/17	1492169946542			New
75%	Pump and Dump	03/30/17	1490869323621	PDZ	Equity	New

Figure 5: Alert tab

Employees tab

The **Employees** tab displays the top 50 employees sorted by their risk score. Only employees with a positive risk score value are displayed. The risk score of an employee is based on their past and currently active alerts. If an employee does not have any alerts in the past 90 days and does not have any currently active alerts, they will not appear in this list.

Figure 6: Employees tab

The Alert Details page shows the basic information about the alert in the header region of the page, and then more information on the tabs of the page.

The **Overview** tab shows the reasoning graph and the associated evidences that created the alert. You can change the start and end dates of the reasoning graph can be changed to show the change in the reasoning over time.



Alert Network tab

The **Network** tab shows the network of communications that were analyzed from the electronic communications. The nodes on the network chart are entities such as a person, an organization, or a ticker. The links between the nodes represent a communication between the entities. You can click the link to show the email that were proof of the communication.

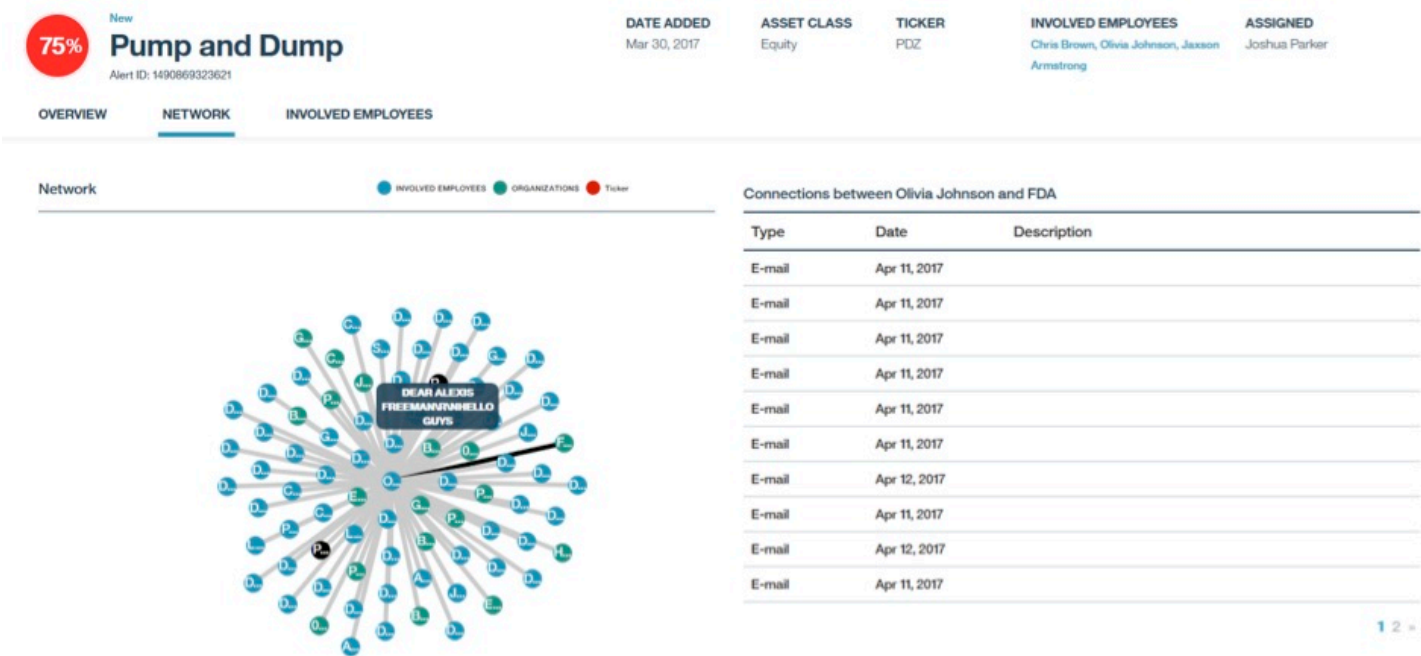


Figure 8: Alert Network tab

Involved Employees tab

The **Involved Employees** tab displays a list of employees that are involved in an alert. You can click an employee to display information about that employee, such as personal information, history, anomalies, and emotion analysis.

The personal information shows the location details, contact information, supervisor details, an alert summary, and a risk score for the employee. The risk score is the overall risk score that is associated with the employee based on their current and past alerts. It is not the risk score that is associated with a specific alert.

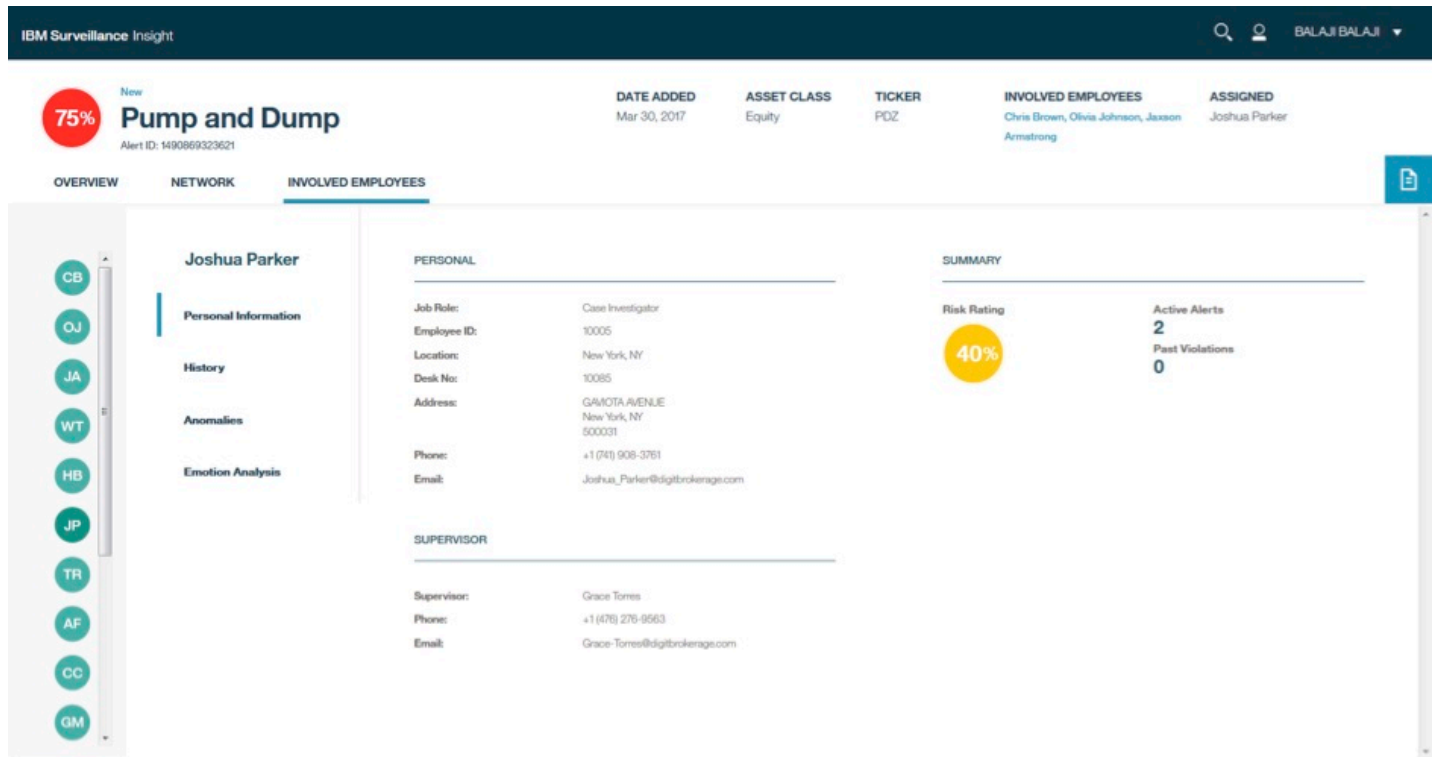


Figure 9: Personal Information

The **History** tab shows the history of current and past alerts for the employee.

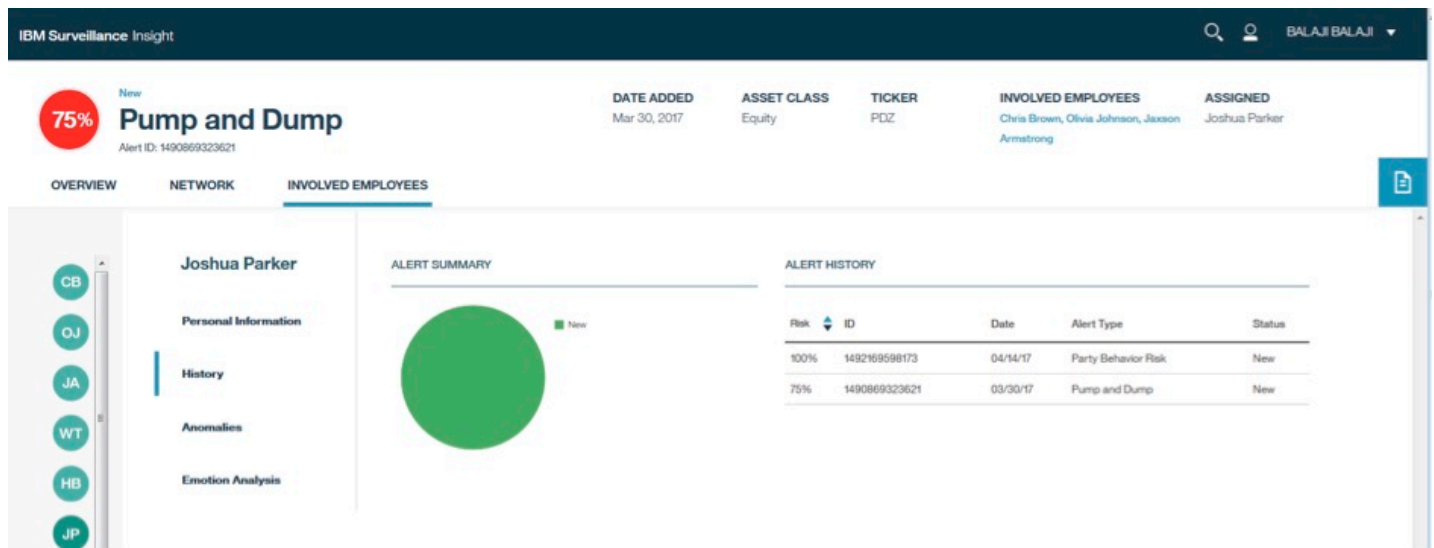


Figure 10: History

The **Anomalies** tab shows the behavior anomalies of the selected employee. Each anomaly has a risk score and a date that is associated with it. These factors determine the position of the circle in the chart. The color of the chart represents the type of anomaly. The data that is used to plot the chart is determined by the start and end dates of the alert.

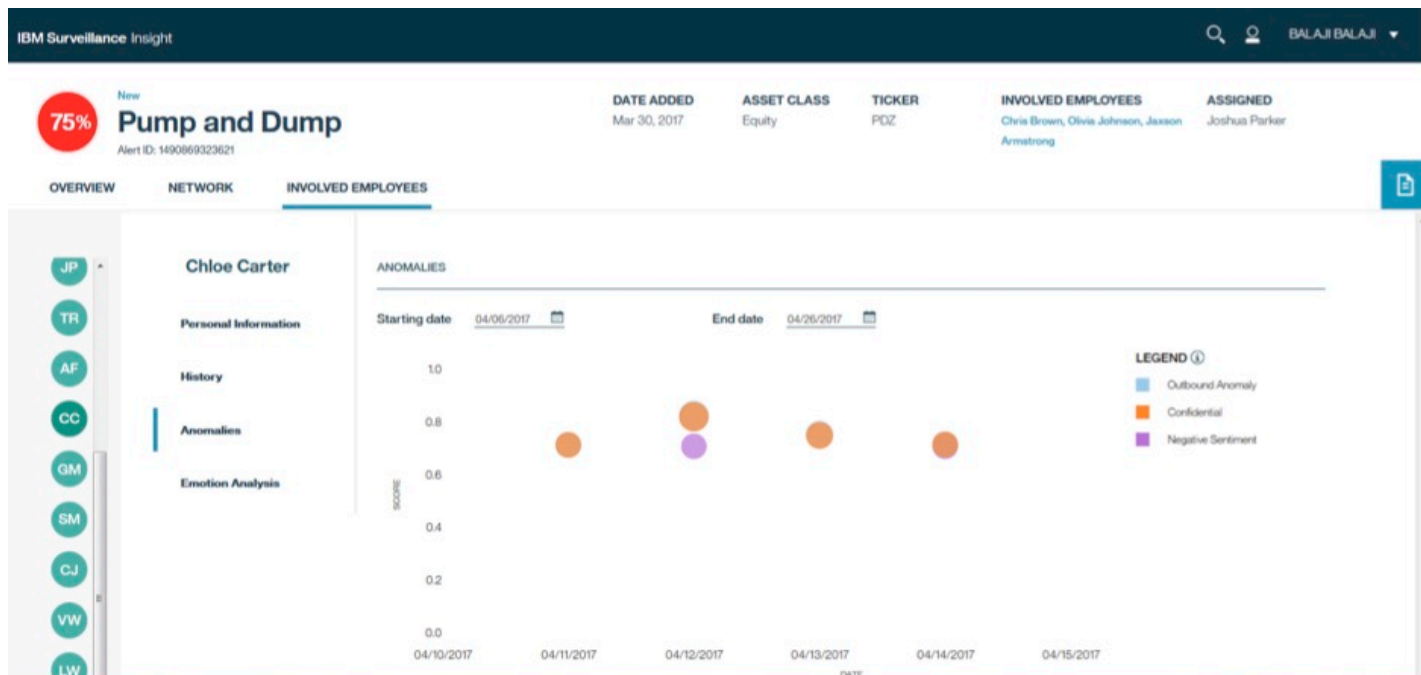


Figure 11: Anomalies tab

The **Emotion Analysis** tab shows the emotional behavior that is portrayed by an employee based on their communications. The chart displays a circle for each instance where the employee's emotion score crosses a threshold. You can click the circle to display a list of communication evidences that contain that specific emotion. The data that is used to plot the chart is determined by the start and end dates of the alert.

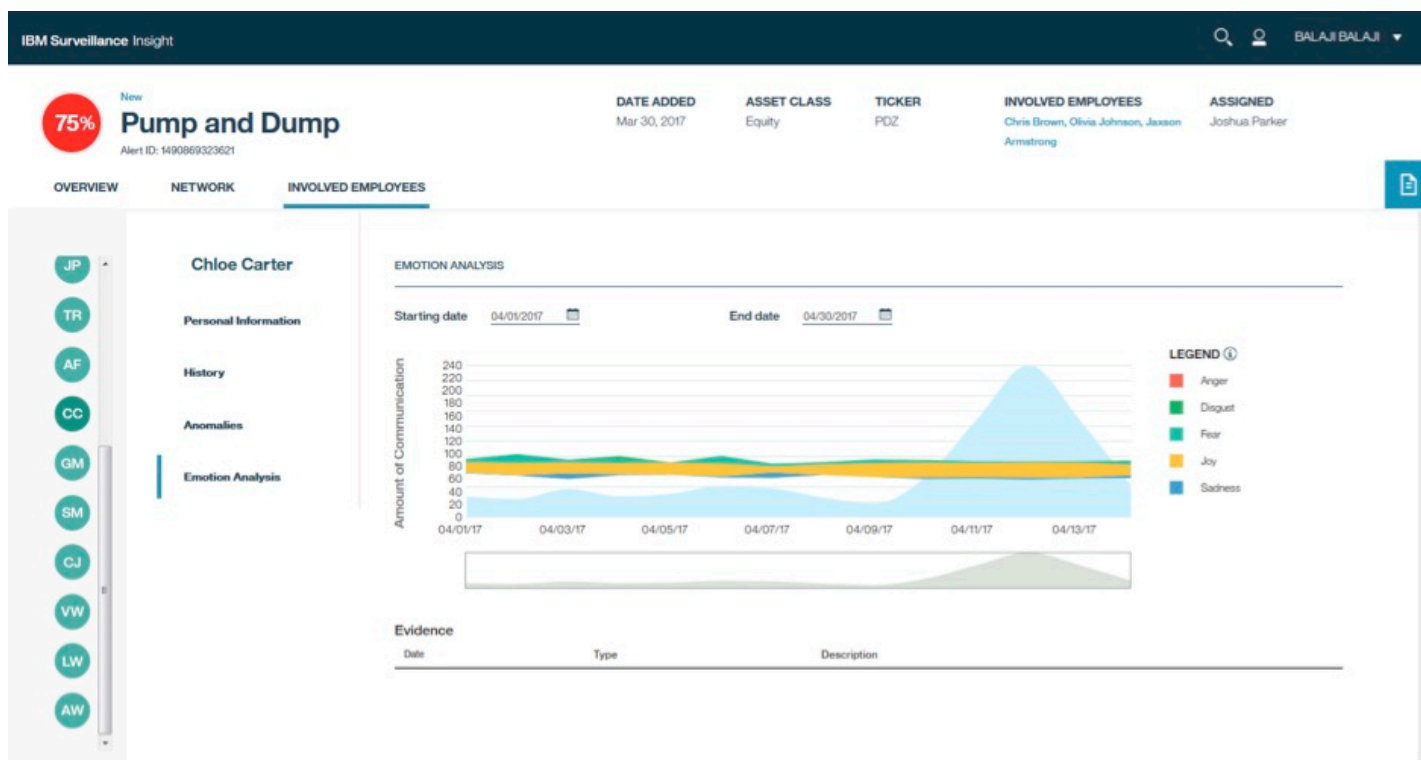


Figure 12: Emotion Analysis tab

Employee Details page

The **Employee Details** page shows the same information as the Involved Employees section of the **Alert** page.

The only difference is that the anomalies chart and the emotional analysis chart use the last 10 days of available data in the database. Whereas, the start and end dates of the alert are used in the **Alert** page.

For more information about the content, see the [“Involved Employees tab”](#) on page 7.

Notes page

Alert investigators can add notes and attach files to an alert from the **Notes** page. You can view the **Notes** page from any of the pages in the **Alert Details** view.

View notes

Click the note icon to view the notes for an alert.

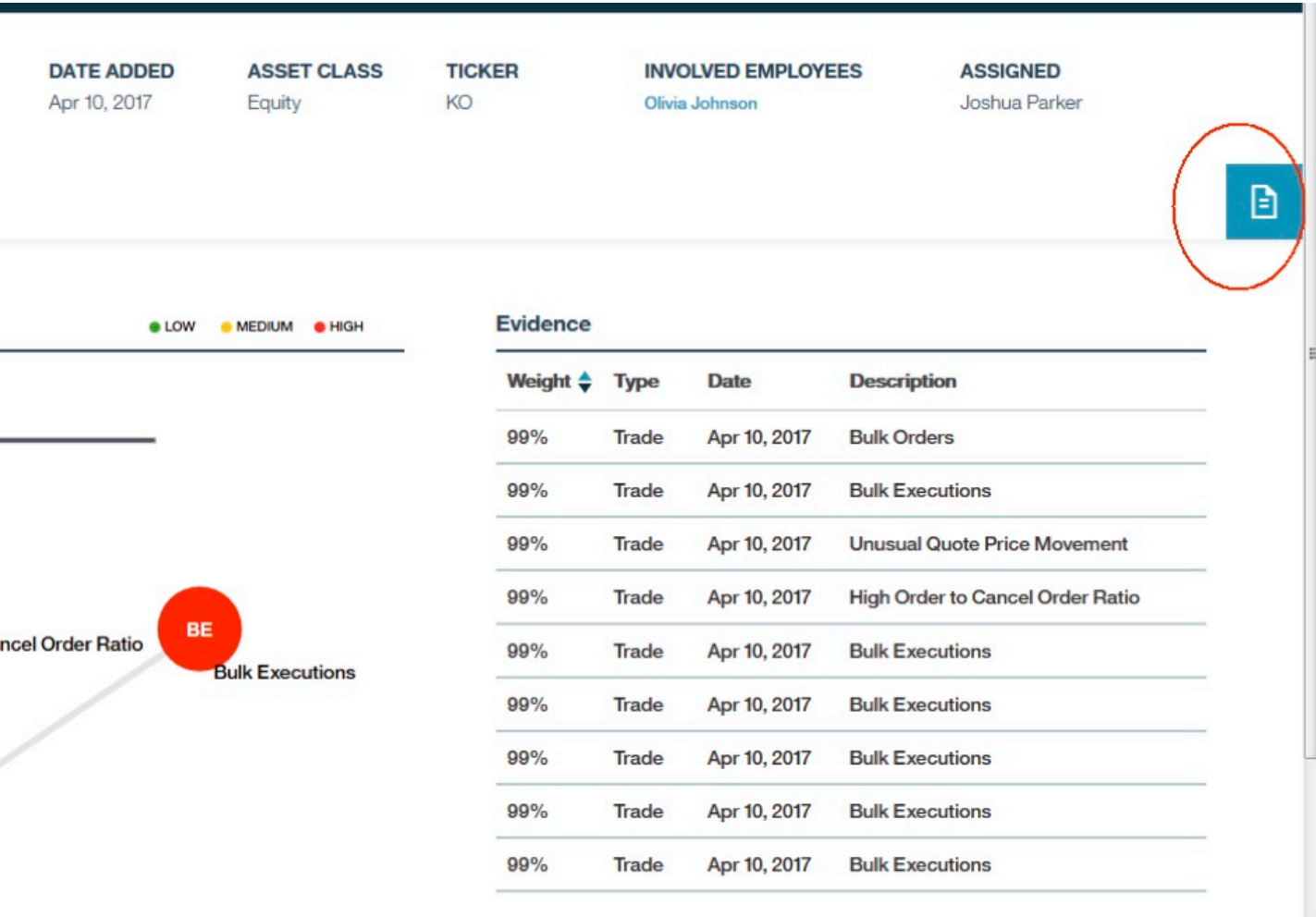


Figure 13: View notes

TICKER

KO

INVOLVED EMPLOYEES

Olivia Johnson

ASSIGNED

Joshua Parker

Evidence

[illegible]

Notes

+ Notes

 Screenshot

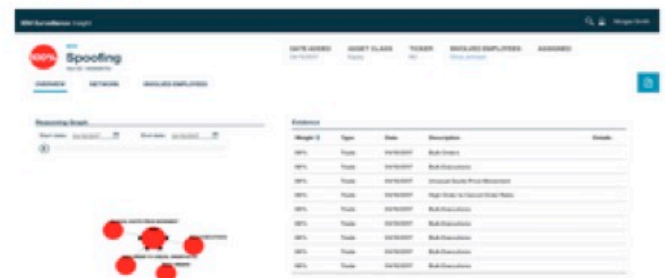
Screenshot-2017-04-20



Additional notes added.

Sadie Ross Apr 20, 2017 at 15:51 pm 

2017-04-07



[Go to Note Summary](#)

Figure 14: Displaying notes

Create notes

From Figure 14 on page 11 page, click **Notes** to add a note. You can also click **Screenshot** to add a screen capture of the current screen.

TICKER
KO

INVOLVED EMPLOYEES
Olivia Johnson

ASSIGNED
Joshua Parker

Evidence

Weight	Type	Date
99%	Trade	Apr 10, 2017
99%	Trade	Apr 10, 2017
99%	Trade	Apr 10, 2017
99%	Trade	Apr 10, 2017
99%	Trade	Apr 10, 2017
99%	Trade	Apr 10, 2017
99%	Trade	Apr 10, 2017
99%	Trade	Apr 10, 2017
99%	Trade	Apr 10, 2017

NotesScreenshot

Cancel

Add

Screenshot-2017-04-20

Spooling

Accounting Weight

High Order to Cancel Order Rate

High Order to Cancel Order Rate

High Order to Cancel Order Rate

High Order to Cancel Order Rate

99% Trade Apr 10 2017

99% Trade Apr 10 2017

99% Trade Apr 10 2017

99% Trade Apr 10 2017

99% Trade Apr 10 2017

99% Trade Apr 10 2017

99% Trade Apr 10 2017

99% Trade Apr 10 2017

99% Trade Apr 10 2017

Go to Note Summary

Figure 15: Create notes

Update notes

You can click **Edit** to change or update a note.

Delete notes

To delete a note, click **Go to Note Summary**, and delete the note.

Notes summaries

You can save a notes summary to a PDF file. To generate a summary, click **Go to Note Summary**, and click **Generate PDF**.

Note actions

The alert investigator can agree or disagree with the notes on the **Note Summary** page. This updates the status of the note in the system.

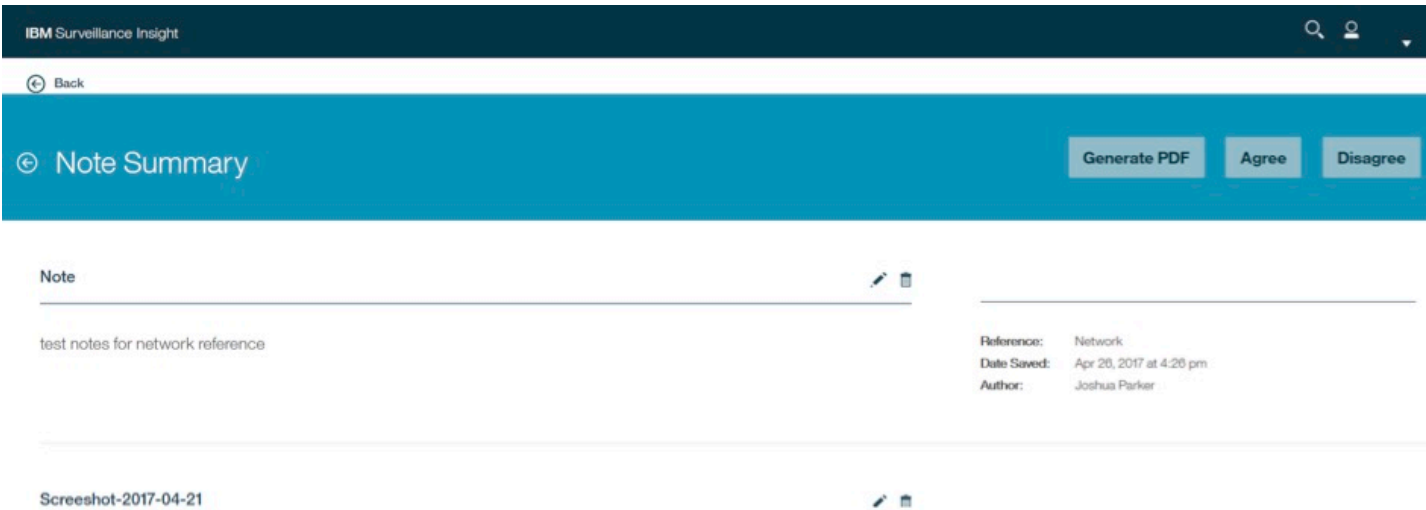


Figure 16: Note Summary page

Search pages

You can search for alerts, employees, and communication types.

Alert Search

You can search for an alert by different criteria, such as by date, alert type, employee, ticker, and status. After you select your criteria, click **Apply** to display the results.

IBM Surveillance Insight

Back

ALERT SEARCH

EMPLOYEE SEARCH

COMMUNICATION SEARCH

Filter by:

Time Range

to

Alert Type

Alert Type

Employee Name

Search name

Ticker

KO

Alert ID

Search ID

Status

New

In Progress

Automatically Escalated

Manually Escalated

Rejected

Closed

Clear

Apply

Score	Type	Date	Status	ID	Asset Class	Ticker
100%	spoofing	Apr 10, 2017	New	1490696753	Equity	KO

Figure 17: Alert Search page

Employee Search

You can search for an employee by their name, ID, location, or role.

Back

ALERT SEARCH

EMPLOYEE SEARCH

COMMUNICATION SEARCH

Filter by:

Employee Name

Search Name

Employee ID

Search ID

Location

Search location

Role

Trader

Banker

Clear

Apply

JS

Jon Snow

Trader, New York, NY

Risk rating: 95%

ID: user1

Active Alerts: 4

Past Violations: 13

View Profile

DW

David Williams

Banker, Boston, MA

Risk rating: 15%

ID: 1005

Active Alerts: 1

Past Violations:

View Profile

JS

Jon Snow

Trader, San Francisco, CA

Risk rating: 65%

ID: 1002

Active Alerts: 1

Past Violations: 5

View Profile

DW

David Williams

Banker, Boston, MA

Risk rating: 15%

ID: 1007

Active Alerts: 1

Past Violations:

View Profile

AS

Arvin Sabre

Trader, Atlanta, GA

Risk rating: 75%

ID: 1004

Active Alerts: 3

Past Violations: 10

View Profile

DT

Dylan Trow

Trader, Atlanta, GA

Risk rating: 71%

ID: 1003

Active Alerts: 2

Past Violations: 6

View Profile

JS

Jon Snow

Trader, New York, NY

Risk rating: 65%

ID: 1002

Active Alerts: 1

Past Violations: 5

View Profile

RN

Rafel Nadal

Banker, Natick, MA

Risk rating: 15%

ID: 1005

Active Alerts: 1

Past Violations:

View Profile

WT

William Thomas

Trader, New York, NC

Risk rating: 71%

ID: 1003

Active Alerts: 2

Past Violations: 6

View Profile

Figure 18: Employee Search page

14 IBM Surveillance Insight for Financial Services Version 2.0.0 : IBM Surveillance Insight for Financial Services Solution Guide

Communication Search

The **Communication Search** allows you to search by communication type, by the people involved in the communication, by the entities, and by the emotions detected in the communication.

Back

ALERT SEARCHEMPLOYEE SEARCHCOMMUNICATION SEARCH

Filter by:

Time Range

to

Channel

Email

Voice

Chat

Employee Name

Search Name

Entity

Search entity

Alert ID

Search ID

Emotion

And/or

Employee	Channel	Date	Description
Robert Brown	E-mail	May 12, 2017	ABCC has landed
Chloe Carter	E-mail	Apr 30, 2017	
Joshua Parker	E-mail	Apr 30, 2017	
Gabriella Myers	E-mail	Apr 30, 2017	
Olivia Johnson	E-mail	Apr 30, 2017	
Gabriella Myers	E-mail	Apr 30, 2017	
Liam Wilson	E-mail	Apr 30, 2017	
Alexis Freeman	E-mail	Apr 29, 2017	
Chloe Carter	E-mail	Apr 29, 2017	
Alexis Freeman	E-mail	Apr 29, 2017	
Gabriella Myers	E-mail	Apr 28, 2017	
Liam Wilson	E-mail	Apr 28, 2017	
Gabriella Myers	E-mail	Apr 27, 2017	
Violet Wagner	E-mail	Apr 27, 2017	
Samantha Morgan	E-mail	Apr 27, 2017	

Figure 19: Communication Search page

Surveillance Insight Workbench 15

Chapter 3. Trade surveillance

IBM Surveillance Insight for Financial Services trade surveillance offers mid and back office surveillance on market activities and communication in order to detect and report possible market abuse activity.

The trade component monitors trade data, detects suspicious patterns against the predefined risk indicators, and reports the patterns. The trade data includes order data, trade data, quotes, executions, and end of the day summary data.

The risk indicators are analyzed by the inference engine. The inference engine uses a risk model to determine whether an alert needs to be created.

Two use cases are provided:

- Pump-and-dump
- Spoofing

The following trade-related risk indicators are available in the Surveillance Insight for Financial Services master data:

- Bulk orders
- High order to order cancel ratio
- Bulk executions
- Unusual quote price movement
- Pump in the stock
- Dump in the stock

Data ingestion

Market data, such as trade, order, quote, and execution data, are uploaded to the Hadoop file system (HDFS) by using the HDFS terminal. The file and folder naming conventions are as follows:

- /user/sifsuser/trade/Trade_<yyyy-mm-dd>.csv
- /user/sifsuser/order/Order_<yyyy-mm-dd>.csv
- /user/sifsuser/execution/Execution_<yyyy-mm-dd>.csv
- /user/sifsuser/quote/Quote_<yyyy-mm-dd>.csv
- /user/sifsuser/EOD/EOD_<yyyy-mm-dd>.csv

The current implementation of the trade use cases expects that there is one file of each type for each day.

The IBM InfoSphere® Streams data loader job monitors the folders. The job reads any new file that is dropped into the folder and sends it for downstream processing.

The following diagram shows the end-to-end flow of data for the trade use cases.

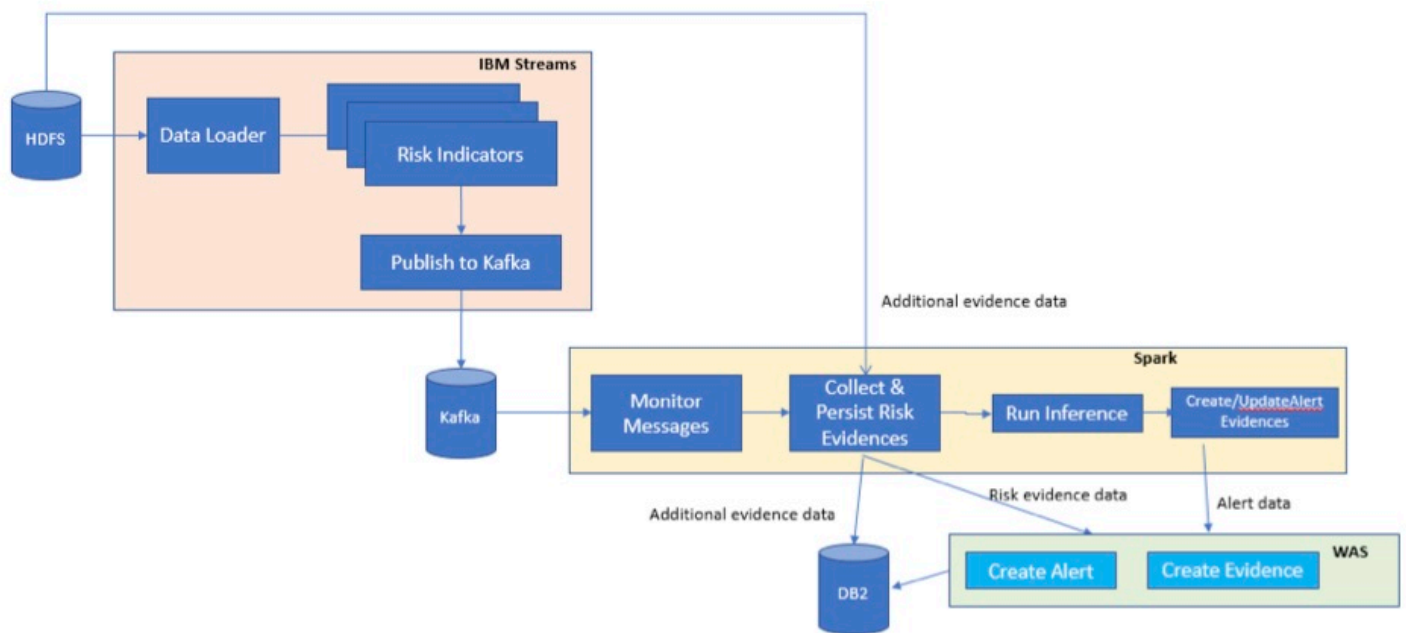


Figure 20: End-to-end data flow for trade use cases

1. Market data is fed into the HDFS, which is monitored by the data loader Streams job.
2. Risk indicators that are identified by the Streams job are passed to a Spark job for downstream processing. This is done through a Kafka topic that is specific to each use case.
3. The Spark job receives messages from the Kafka topic and (optionally) collects more evidence data than might be required. (This additional data is not present in the message that is coming from the Streams job.)
4. The risk evidences are created in the Surveillance Insight database through the Create Evidence REST service.
5. Any additional data that might be needed is populated in the database by directly connecting to the database.
6. The inference engine is invoked to determine the alert creation based on the available risk evidences.
7. Alerts are created or updated in the database based on the outcome of the inference engine. This is done through the Create Alert REST service.

Trade Toolkit data schemas

The following are the Trade Toolkit data schemas.

Ticker price schema

symbol,datetime,price

Table 1: Ticker price schema		
Field name	Field type	Description
symbol	String	The ticker corresponding to the trade
datetime	String	The date and time at which the trade occurred
price	Float	The unit price of the stocks traded

Execution schema

Id, Symbol, Datetime, Brokerid, Traderid, Clientid, effectiveTime, expireTime, timeInForce, exposureDuration, tradingSession, tradingSessionSub, settleType, settleDate,

Currency, currencyFXRate, execType, trdType, matchType, Side, orderQty, Price, exchangeCode, refQuoteId, refOrderId

For more information about the fields in this schema, refer to the [FIX wiki](http://fixwiki.org/fixwiki/ExecutionReport/FIX.5.0SP2%2B) (<http://fixwiki.org/fixwiki/ExecutionReport/FIX.5.0SP2%2B>)

Table 2: Execution schema		
Field name	Field type	Description
Id	String	Unique identifier for the execution
Symbol	String	The ticker corresponding to the trade
Datetime	String	The date and time at which the trade occurred. The format is yyyy-mm-dd hh:mm:ss
Brokerid	String	The ID of the broker that is involved in this execution
Traderid	String	The ID of the trader that is involved in this execution
Clientid	String	The ID of the client that is involved in this execution
effectiveTime	String	The date and time stamp at which the execution is effective
expireTime	String	The date and time stamp when this execution will expire
timeInForce	String	Specifies how long the order remains in effect. Absence of this field is interpreted as DAY
exposureDuration	String	The time in seconds of a "Good for Time" (GFT) TimeInForce
tradingSession	String	Identifier for a trading session
tradingSessionSub	String	Optional market assigned sub identifier for a trading phase within a trading session
settlType	String	Indicates order settlement period. If present, SettlDate overrides this field. If both SettlType and SettlDate are omitted, the default for SettlType is 0 (Regular)
settlDate	String	Specific date of trade settlement (SettlementDate) in YYYYMMDD format
Currency	String	The currency in which the execution price is represented
currencyFXRate	Float	The foreign exchange rate that is used to calculate SettlCurrAmt from Currencyto SettlCurrency

Table 2: Execution schema (continued)		
Field name	Field type	Description
execType	String	Describes the specific ExecutionRpt (for example, Pending Cancel) while OrdStatus will always identify the current order status (for example, Partially Filled)
trdType	String	Type of trade
matchType	String	The point in the matching process at which this trade was matched
Side	String	Denotes BUY or SELL execution
orderQty	Int	The volume that is fulfilled by this execution
Price	Float	The price per unit for this execution
exchangeCode	String	
refQuoteId	String	The quote that corresponds to this execution
refOrderId	String	Refers to the order corresponding to this execution

Order schema

Id, Symbol, Datetime, effectiveTime, expireTime, timeInForce, exposureDuration, settlType, settlDate, Currency, currencyFXRate, partyId, orderType, Side, orderQty, minQuantity, matchIncr, Price, manualOrderIndicator, refOrderId, refOrderSource

For more information about the fields in this schema, refer to the [FIX wiki](http://fixwiki.org/fixwiki/ExecutionReport/FIX.5.0SP2%2B) (<http://fixwiki.org/fixwiki/ExecutionReport/FIX.5.0SP2%2B>)

Table 3: Order schema		
Field name	Field type	Description
Id	String	Unique identifier for the order
Symbol	String	The ticker corresponding to the trade
Datetime	String	The date and time at which the order was placed. The format is yyyy-mm-dd hh:mm:ss
effectiveTime	String	The date and time stamp at which the order is effective
expireTime	String	The date and time stamp when this order will expire
timeInForce	String	Specifies how long the order remains in effect. If this value is not provided, DAY is used as the default
exposureDuration	String	The time in seconds of a "Good for Time" (GFT) TimeInForce

Table 3: Order schema (continued)		
Field name	Field type	Description
settlType	String	Indicates order settlement period. If present, SettlDate overrides this field. If both SettlType and SettlDate are omitted, the default for SettlType is 0 (Regular)
settlDate	String	Specific date of trade settlement (SettlementDate) in YYYYMMDD format
Currency	String	The currency in which the order price is represented
currencyFXRate	Float	The exchange rate that is used to calculate the SettlCurrAmt from Currency to SettlCurrency
partyId	String	The trader that is involved in this order
orderType	String	CANCEL represents an order cancellation. Used with refOrderId.
Side	String	Indicates a BUY or SELL order
orderQty	Int	The order volume
minQuantity	Int	Minimum quantity of an order to be executed
matchIncr	Int	Allows orders to specify a minimum quantity that applies to every execution (one execution might be for multiple counter-orders). The order can still fill against smaller orders, but the cumulative quantity of the execution must be in multiples of the MatchIncrement
Price	Float	The price per unit for this order
manualOrderIndicator	boolean	Indicates whether the order was initially received manually (as opposed to electronically) or if it was entered manually (as opposed to it being entered by automated trading software)
refOrderId	String	Used with the orderType. Refers to the order that is being canceled
refOrderSource	String	The source of the order that is represented by a cancellation order

Quote schema

Id, Symbol, Datetime, expireTime, exposureDuration, tradingSession, tradingSessionSub, settlType, settlDate, Currency, currencyFXRate, partyId, commPercentage, commType, bidPrice, offerPrice, bidSize, minBidSize, totalBidSize, bidSpotRate, bidFwdPoints, offerSize, minOfferSize, totalOfferSize, offerSpotRate, offerFwdPoints

For more information about the fields in this schema, refer to the [FIX wiki](http://fixwiki.org/fixwiki/ExecutionReport/FIX.5.0SP2%2B) (<http://fixwiki.org/fixwiki/ExecutionReport/FIX.5.0SP2%2B>)

Table 4: Quote schema		
Field name	Field type	Description
Id	String	Unique identifier for the quote
Symbol	String	The ticker corresponding to the trade
Datetime	String	The date and time at which the quote was placed. The format is yyyy-mm-dd hh:mm:ss
expireTime	String	The date and time stamp when this quote will expire
exposureDuration	String	The time in seconds of a "Good for Time" (GFT) TimeInForce
tradingSession	String	Identifier for a trading session
tradingSessionSub	String	Optional market assigned sub identifier for a trading phase within a trading session
settlType	String	Indicates order settlement period. If present, SettlDate overrides this field. If both SettlType and SettlDate are omitted, the default for SettlType is 0 (Regular)
settlDate	String	Specific date of trade settlement (SettlementDate) in YYYYMMDD format
Currency	String	The currency in which the quote price is represented
currencyFXRate	Float	The exchange rate that is used to calculate SettlCurrAmt from Currency to SettlCurrency
partyId	String	The trader that is involved in this quote
commPercentage	Float	Percentage of commission
commType	String	Specifies the basis or unit that is used to calculate the total commission based on the rate
bidPrice	Float	Unit price of the bid
offerPrice	Float	Unit price of the offer
bidSize	Int	Quantity of bid
minBidSize	Int	Type of trade
totalBidSize	Int	
bidSpotRate	Float	Bid F/X spot rate
bidFwdPoints	Float	Bid F/X forward points added to spot rate. This can be a negative value
offerSize	Int	Quantity of the offer
minOfferSize	Int	Specifies the minimum offer size
totalOfferSize	Int	

Table 4: Quote schema (continued)		
Field name	Field type	Description
offerSpotRate	Float	Offer F/X spot rate
offerFwdPoints	Float	Offer F/X forward points added to spot rate. This can be a negative value

Trade schema

Id, Symbol, Datetime, Brokerid, Traderid, Clientid, Price, Volume, Side

Table 5: Trade schema		
Field name	Field type	Description
Id	String	Unique identifier for the trade
Symbol	String	The ticker corresponding to the trade
Datetime	String	The date and time at which the trade occurred. The format is yyyy-mm-dd hh:mm:ss
Brokerid	String	The id of the broker involved in the trade
Traderid	String	The id of the trader involved in the trade
Clientid	String	The id of the client involved in the trade
Price	Float	The unit price of the stocks traded
Volume	Int	The volume of stocks traded
Side	String	The BUY or SELL side of the trade

End of day (EOD) schema

Id, Symbol, Datetime, openingPrice, closingPrice, dayLowPrice, dayHighPrice, Week52LowPrice, Week52HighPrice, marketCap, totalVolume, industryCode, div, EPS, beta, description

Table 6: End of day (EOD) schema		
Field name	Field type	Description
Id	String	Unique identifier for the trade
Symbol	String	The ticker corresponding to the trade
Datetime	String	The date and time at which the trade occurred. The format is yyyy-mm-dd hh:mm:ss
openingPrice	Float	The opening price of the ticker for the date that is specified in the datetime field
closingPrice	Float	The closing price of the ticker for the date that is specified in the datetime field
dayLowPrice	Float	The lowest traded price for the day for this ticker

Table 6: End of day (EOD) schema (continued)		
Field name	Field type	Description
dayHighPrice	Float	The highest traded price for the day for this ticker
Week52LowPrice	Float	The 52-week low price for this ticker
Week52HighPrice	Float	The 52-week high price for this ticker
marketCap	Float	The market cap for this ticker
totalVolume	Int	The total outstanding volume for this ticker as of today
industryCode	String	The industry to which the organization that is represented by the ticker corresponds to
div	Float	
EPS	Float	
beta	Float	
description	String	The description of the organization that is represented by the ticker

Event schema

id, eventType, startTime, windowSize, traderId, symbol, data

Table 7: Event schema		
Field name	Field type	Description
id	String	System generated id for the event
eventType	String	The type of the event
startTime	String	The system time when the event occurred
windowSize	Float	The size (in seconds) of the data window that the operator used while looking for events in the input data stream.
traderId	String	The trader id associated with the event
symbol	String	The symbol associated with the event
data	List of event data	Event specific data list. See Event Data schema

Event data schema

name, value

Table 8: Event data schema		
Field name	Field type	Description
name	String	The name of the event property
value	String	The value of the event property

Audio metadata schema

partyid, initiator_phone_num, initiator_device_id, participants_partyid, participants_phone_num, participants_device_id, voice_file_name, date, callstarttime, callendtime, analysis_policy_id, global_comm_id

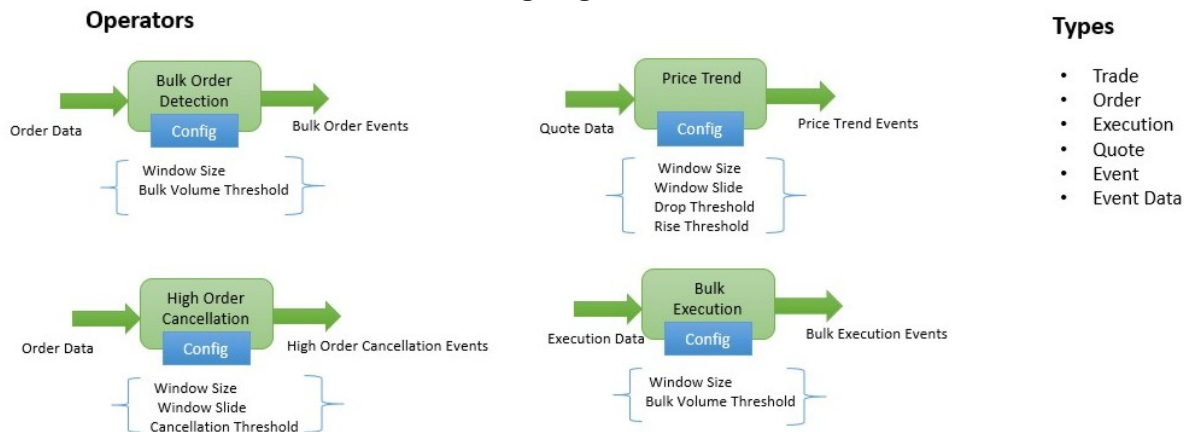
Table 9: Audio metadata schema

Field name	Field type	Description
partyid	String	Call Initiator's Party id
initiator_phone_num	String	Phone number of call Initiator
initiator_device_id	String	Device id from which call was initiated
participants_partyid	String	Partyid of receiving participants. Multiple values are separated by ;
participants_phone_num	String	Phone number of receiving participants. Multiple values are separated by ;
participants_device_id	String	Device id of receiving participants. Multiple values are separated by ;
voice_file_name	String	Audio file name that needs to be processed.
date	String	Date the call was recorded in YYYY-MM-DD format
callstarttime	String	Call start time in hh:mm:ss format
callendtime	String	Call end time of call in hh:mm:ss format
analysis_policy_id	String	Policy ID that should be applied while analyzing this audio
global_comm_id	String	Unique global communication id attached to this audio

Trade Surveillance Toolkit

The Trade Surveillance Toolkit helps the solution developers to focus on specific use case development.

The toolkit contains basic data types, commonly used functional operators relevant to trade analytics, and adapters for some data sources, as shown in the following diagram.



For information about the schemas for the types that are defined in the toolkit, see [“Trade Toolkit data schemas”](#) on page 18.

Bulk Order Detection operator



Purpose

Looks at a sliding window of orders and checks if total order volume is over the Bulk Volume Threshold. It is grouped by trader, ticker, and order side (buy/sell). The sliding window moves by 1 second for every slide.

Input

Order Data according to the schema.

Output event contents

Id: unique ID for this event.
Event Time: time in input data, not system time.
Event Type: BULK_ORDER.
Trader ID: ID of the trader who is placing the order.
Ticker
Event Data.
orderQty: total volume of orders in the window for Trader ID.
Side: BUY or SELL.
maxOrderPrice: maximum order price that was seen in the current window.

Configuration

Window Size: time in seconds for collecting data to analyze.
Bulk Volume Threshold: Volume threshold that is used to trigger events.

High Order Cancellation operator



Purpose

Looks at a sliding window of window Size (in seconds) and checks if total order volume to order cancellation volume for a trader is above the Cancellation Threshold. It is grouped by trader, ticker, and order side (buy/sell).

Input

Order Data according to the schema.

Output event contents

Id: unique ID for this event.

Event Time: time in input data, not system time.
Event Type: HIGH_CANCEL_RATIO
Trader ID: ID of the trader who is placing the order.
Ticker
Event Data.
Side: BUY or SELL.
ratio: order volume versus cancellation ratio.

Configuration

Window Size: time in seconds for collecting data to analyze.
Window Slide: Slide value for the window in seconds.
Cancellation Threshold: Volume threshold that is used to trigger events.

Price Trend operator



Purpose

Looks at a sliding window of quotes and computes the rise or drop trend (slope) for offer and bid prices. It fires an event if the price slope rises above the Rise Threshold or drops below the Drop Threshold. The former indicates an unusual rise in the quotes and the latter indicates an unusual drop in the quotes. The analysis is grouped by ticker.

Input

Quote Data according to the schema.

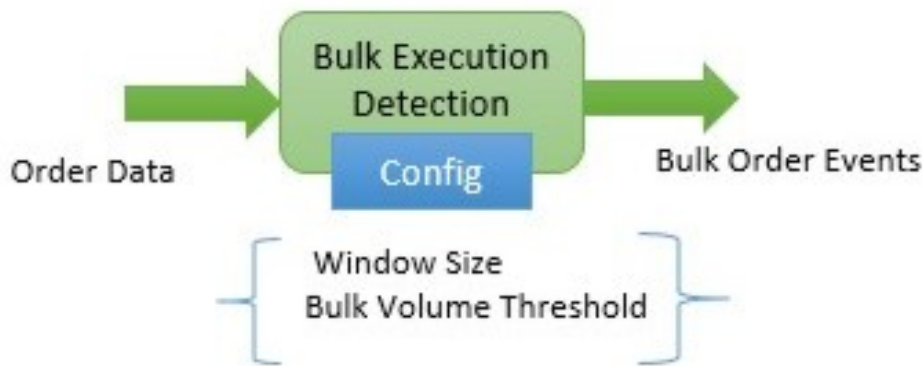
Output event contents

Id : unique ID for this event.
Event Time: time in input data, not system time.
Event Type: PRICE_TREND.
Trader ID: not applicable.
Ticker
Event Data.
Side: BID or OFFER.
Slope: slope of the bid or offer price.

Configuration

Window Size: time in seconds for collecting data to analyze.
Window Slide: Slide value for the window in seconds.
Drop Threshold: Threshold that indicates an unusual downward trend in the quotes.
Rise Threshold: Threshold that indicates an unusual rise trend in the quotes.

Bulk Execution Detection operator



Purpose

Looks at a sliding window of executions and checks if the total executed volume is above the Bulk Volume Threshold. It is grouped by trader, ticker, and order side (buy/sell). The sliding window moves by 1 second for every slide.

Input

Execution Data according to the schema.

Output event contents

Id : unique ID for this event.
Event Time: time in input data, not system time.
Event Type: BULK_EXEC
Trader ID: ID of the trader who is placing the order.
Ticker
Event Data:
orderQty: total volume of executions in the window for Trader ID.
Side: BUY or SELL.
TotalExecValue: price * execution quantity for this window. It is grouped by ticker, trader, and side.

Configuration

Window Size: time in seconds for collecting data to analyze.
Bulk Volume Threshold: The volume threshold that is used to trigger events.

Pump-and-dump use case

The solution contains a pump-and-dump use case, which carries out structured analysis of trade, order, and execution data and unstructured analysis of email data. The result is a daily score for the pump-and-dump indication.

The pump-and-dump score is distributed daily among the top five traders. Top five is determined based on the positions that are held by the traders.

Triggering the pump-and-dump rules

Ensure that the following folders exist on the Hadoop file system. The folders are:

- /user/sifsuser/trade/
- /user/sifsuser/order/
- /user/sifsuser/execution/
- /user/sifsuser/quote/
- /user/sifsuser/EOD/
- /user/sifsuser/sifsdata/ticker_summary/ticker_summary/
- /user/sifsuser/sifsdata/position_summary/
- /user/sifsuser/sifsdata/positions/
- /user/sifsuser/sifsdata/pump_dump/

- `/user/sifuser/sifsdata/trader_scores/`

Both structured market data and unstructured email data are used for pump-and-dump detection. For accurate detection, ensure that you load the email data before you load the structured data. After structured data is pushed into Hadoop, the pump-and-dump implementation processes this data and automatically triggers the inference engine. The inference engine considers evidences from both email and structured data analysis to determine the risk score.

Understanding the pump-and-dump analysis results

When the data is loaded into Surveillance Insight for Financial Services, the pump-and-dump rules are triggered and the following files are created on the Hadoop file system:

- Date-wise trade summary data, including moving averages, is created in `/user/sifuser/sifsdata/ticker_summary/ticker_summary_<date>.csv`
- Date-wise position summary data is created in `/user/sifuser/sifsdata/positions/top5Positions_<date>.csv` and `/user/sifuser/sifsdata/position_summary/position_summary_<date>.csv`
- Date-wise pump-and-dump score data is created in `/user/sifuser/sifsdata/pump_dump/pump_dump_<date>.csv`
- Date-wise trader score data is created in `/user/sifuser/sifsdata/trader_scores/trader_scores_<date>.csv`

The Spark job for pump-and-dump evidence collection is run for each date. This job collects all of the evidences for the day from Hadoop and populates the following tables in the SIFS database:

- Risk_Evidence
- Evidence_Ticker_Rel
- Evidence_Involved_Party_Rel

The Spark job also runs the inference engine, which applies a risk model and detects whether an alert needs to be generated for the evidence. Based on the result, either a new alert is generated, an existing alert is updated, or no action is taken. The alert information is populated to the following tables:

- Alert
- Alert_Ticker_Rel
- Alert_Involved_Party_Rel
- Alert_Risk_Indicator_Score
- Alert_Evidence_Rel

After the evidence and alert tables are updated, the pump-and-dump alert appears in the dashboard.

Pump-and-dump alerts are long running in that they can span several days to weeks or months. The same alert is updated daily if the risk score does not decay to 0.

The following rules explain when an alert is generated versus when an alert is updated:

1. If no evidence of pump-and-dump activity for a ticker from either structured or unstructured analysis exists, or if the risk score is too low, then no alerts are created.
2. If the inference engine determines that an alert must be created, then an alert is created in the Surveillance Insight database against the ticker. The top 5 traders for the day for that ticker are also associated with the alert.
3. After the alert is created, the alert is updated daily with the following information while the ticker remains in a pump or dump state:
 - New alert risk indicator scores are created for each risk indicator that is identified on the current date.
 - The alert end date is updated to the current date.
 - The alert score is updated if the existing score is less than the new score for the day.
 - The new evidences for the day is linked to the existing alert.
 - New parties that are not already on the alert are linked to the alert. New parties would be the top 5 parties for the ticker for the current date.
4. After the alert is create, if the ticker goes into an undecided state, the risk score will start decaying daily. If the score is not 0, the alert is updated as indicated in step 3. For an undecided state, the alert has no pump or dump evidences for the date.

Spoofing detection use case

The spoofing detection use case implementation analyzes market data events and detects spoofing patterns.

A spoofer is a trader who creates a series of bulk buy or sell orders with increasing bid or decreasing ask prices with the intention of misleading the buyers and sellers in a direction that results in a profit for the spoofer. The spoofer cancels the bulk orders before they are completed and then sells or buys the affected stocks at a favorable price that results from the spoofing activity. By analyzing the stock data that is streaming in from the market, the spoofing detection use case detects spoofing activity in near real time.

Triggering the spoofing rules

The spoofing use case implementation requires order, execution, and quote data to detect the spoofing pattern. Loading the data triggers the spoofing rules and updates the alert and score tables in the database.

Understanding the spoofing results

The spoofing use case leverages the Trade Surveillance Toolkit to detect spoofing. It analyzes the market data by looking at the events that are fired by the toolkit and generates alerts if a spoofing pattern is detected. The evidence is then used to determine whether an alert needs to be generated. This decision is made by the inference engine. The alert and the evidence are stored in the Surveillance Insight database by using the REST services.

Spoofing user interface

A spoofing alert appears in the **Alerts** tab.

ALERTS		EMPLOYEES				
Score	Type	Date	ID	Ticker	Asset Class	Status
100%	Spoofing	Apr 10, 2017	1490696753	KO	Equity	New
100%	Party Behavior Risk	Apr 21, 2017	1492750245803			New
100%	Party Behavior Risk	Apr 21, 2017	1492754415558			New
100%	Party Behavior Risk	Apr 21, 2017	1492756957506			New
100%	Party Behavior Risk	Apr 21, 2017	1492758334789			New
100%	Party Behavior Risk	Apr 21, 2017	1492758992979			New
100%	Party Behavior Risk	Apr 21, 2017	1492759954975			New

Figure 21: Spoofing alert

Click the alert to see the alert overview and reasoning.

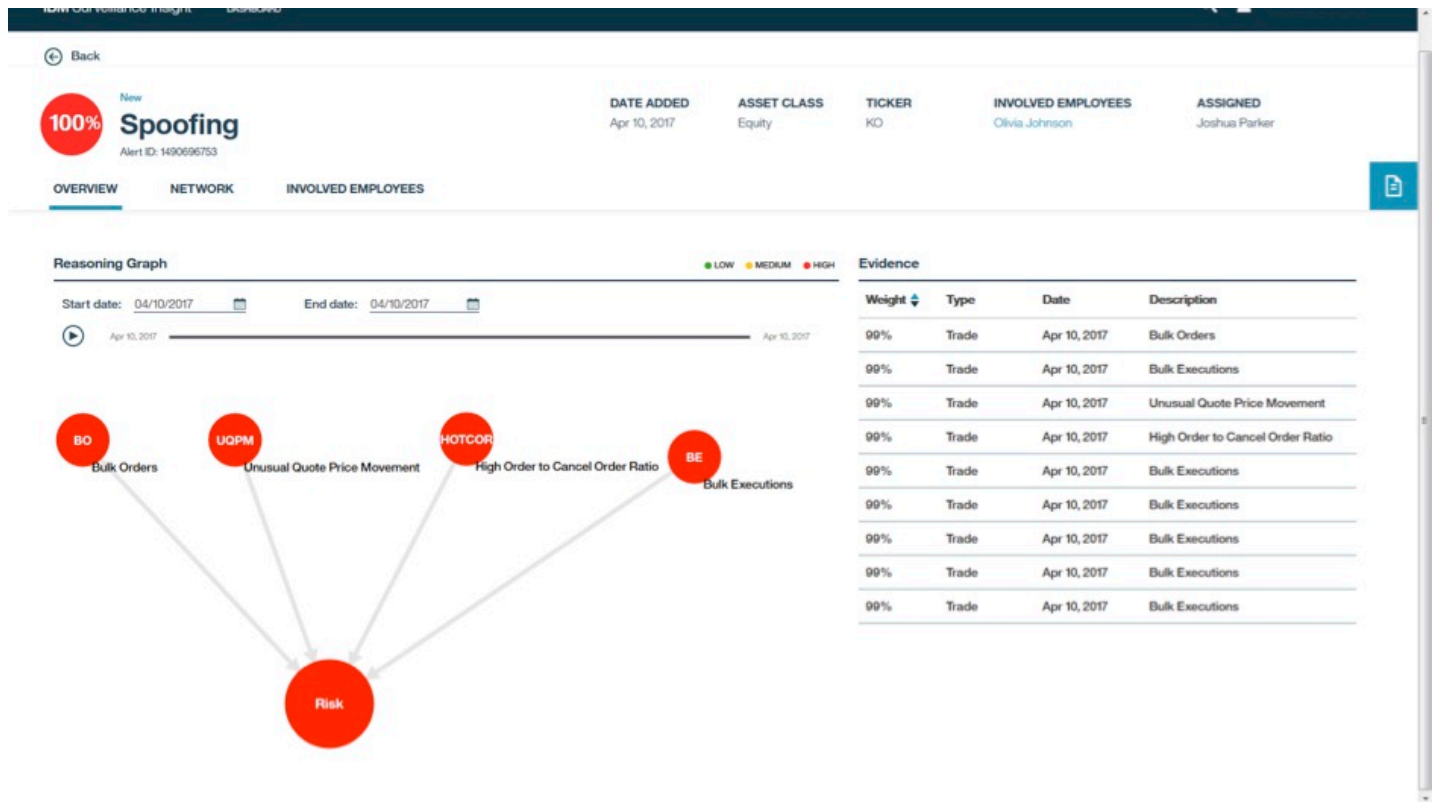


Figure 22: Spoofing overview page

The evidence shows the spoofing pattern where in the bulk orders, unusual quote price movement, and high ratio of orders to cancellation are followed by a series of bulk executions. These evidences contribute to the overall risk as shown in the reasoning graph. In this example, all of the evidences have a 99% weight. This is because for spoofing to happen, each of the events, represented by the risk indicators, should necessarily happen. Otherwise, the pattern would not qualify for spoofing.

Extending Trade Surveillance

Solution developers can use the solution's architecture to develop new trade surveillance use cases.

The Surveillance Insight platform is built around the concept of risk indicators, evidences, and alerts. A use case typically identifies a certain type of risk in terms of risk indicators.

One of the first things for any use case implementation on the platform is to identify the risk indicators that are to be detected by the use case. After the risk indicators are identified, the kinds of evidence for the risk must be identified. For example, the indicators might be trade data or email data that showed signs of risk during analysis.

A risk model must be built that uses the evidence so that the inference engine can determine whether an alert must be generated.

The type of alerts that are generated by the use case must also be identified.

The identified risk indicators, the model, and the alert types are then loaded into the The Surveillance Insight database:

- The risk indicators must be populated in the RISK_INDICATOR_MASTER table.
- The risk model must be populated in the RISK_MODEL_MASTER table.
- The alert type must be populated in the ALERT_TYPE_MASTER table.

End-to-end implementation

The following diagram shows the sequence of steps that are involved in developing a new Surveillance Insight for Financial Services (SIFS) use case:

End to End Process Overview

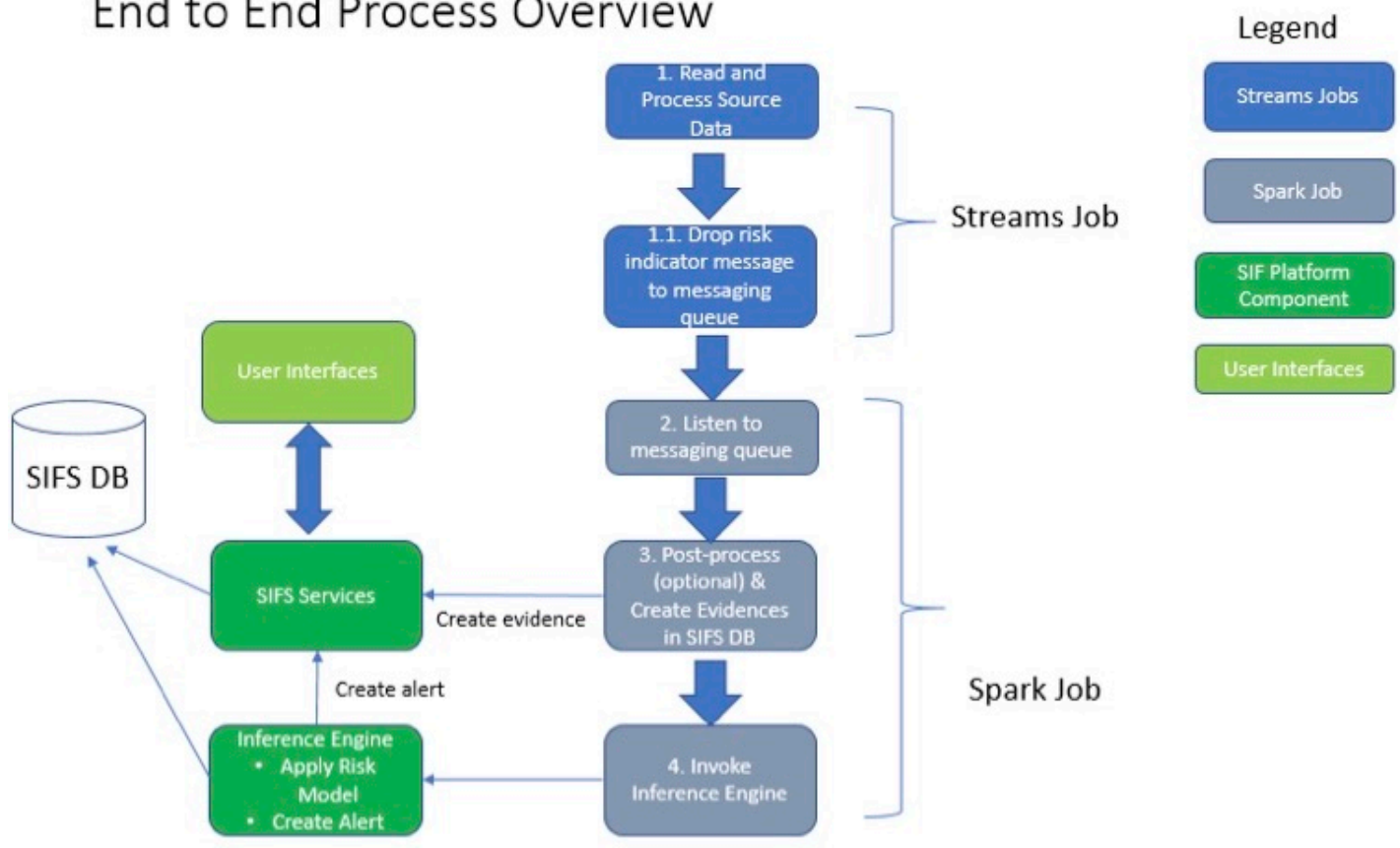


Figure 23: End-to-end implementation for a new use case

1. Read and process the source data.

This step involves implementing the core logic that is involved in reading and analyzing the market data for patterns of interest based on the use case requirements. This step is usually implemented in IBM InfoSphere Streams.

a. Understand the event-based approach that is needed to perform trading analytics.

One of the fundamental principles on which the Trade Surveillance Toolkit operates is generating events that are based on stock market data. The toolkit defines a basic event type that is extensible with event-specific parameters. Different types of stock data, such as orders, quotes, executions, and trade data, are analyzed by the operators in the toolkit. Based on the analysis, these operators generate different types of events.

The primary benefit of the event-based model is that it allows the specific use case implementation to delegate the basic functions to the toolkit and focus on the events that are relevant. Also, this model allows the events to be generated one time and then reused by other use cases. It also drastically reduces the volume of data that the use case must process.



- b. Identify the data types and analytics that are relevant to the use case.

Identify what data is relevant and what analytics need to be performed on the data. These analytic measures are then used to identify the events that are of interest to the use case.

- c. Identify Trading Surveillance Toolkit contents for reuse.

Map the data types and events that are identified to the contents in the toolkit. The result of this step is a list of data types and operators that are provided by the toolkit.

- d. Design the Streams flows by leveraging the toolkit operators.

This step is specific to the use case that you are implemented.

In this step, the placement of the Trading Surveillance Toolkit operators in the context of the larger solution is identified. The configuration parameter values for the different operators are identified. Also, data types and operators that are not already present in the toolkit are designed.

- e. Implement the use case and generate the relevant risk indicators.

- 2. Drop the risk indicator message on a Kafka topic.

The output of step 1 is a series of one or more risk indicators that are found by the data analysis process. The risk indicator data is published to a Kafka topic for downstream processing. A new Kafka topic must be created on the Surveillance Insight platform for this purpose. The topic should be named something such as `sifs.<usecase_name>`. This step is implemented by using InfoSphere Streams or the same technology that was used for implementing step 1.

- 3. Read the risk indicator message from the Kafka topic.

In this step, the message that is dropped on to the Kafka topic is read and sent for downstream processing. This step is typically implemented as a Spark job that processes the risk indicator messages.

- 4. Create the risk evidences in the Surveillance Insight database.

The risk indicator message that is obtained from step 3 is used to create the evidence in the Surveillance Insight database. The `createEvidence` REST service must be run to create the evidence. The JSON input that is used by the `createEvidence` service is prepared by using the risk indicator message that is obtained from step 3.

- 5. Run the inference engine.

The next step is to run the inference engine with the risk evidence details. The inference engine applies the risk model for the use case and determines whether a new alert must be created for the identified risk evidences.

- 6. Create an alert in the Surveillance Insight database.

Based on the results of step 5, this step runs the `createAlert` REST service to create an alert in the Surveillance Insight database.

Chapter 4. E-Comm surveillance

The e-comm component processes unstructured data such as chat and email data. e-comm data is evaluated against various features, and certain risk indicators are computed. These risk indicators are later analyzed by the inference engine to detect alarming conditions and generate alerts. The e-comm component evaluates the features based on the policies that are defined in the system.

Currently the following e-comm-related risk indicators are available in the Surveillance Insight for Financial Services master data:

- Anger anomaly
- Sad anomaly
- Negative sentiment anomaly
- Inbound anomaly
- Outbound anomaly
- Confidential anomaly
- Unusual mail size
- Unusual number of attachments
- Unusual communication timings
- Unusual number of recipients
- Recruit victims
- Recruit co-conspirators
- Efforts on recruit victims
- Efforts on recruit co-conspirators

E-Comm Surveillance Toolkit

The E-comm Surveillance Toolkit helps the solution developers to focus on specific use case development.

Exact behavior operator

This Java™ based feature extraction operator invokes the emotion/sentiment library to detect if any emotions or sentiments are displayed in the communication content. The operator invokes the library if the `executeFlag` of the emotion feature is set to `true`.

Input type

FeatureInOut

Output type

FeatureInOut

If the library is invoked and returns Success, the JSON formatted response of the library is populated in the `featureOutput` element of the emotion feature.

Configuration parameters

`dictPath`—Path of emotion library dictionary.

`rulePath`—Path of emotion library rule file.

Concept mapping operator

This Java based feature extraction operator invokes the concept mapper library to detect if any recruit victims or recruit conspirators are reported in the communication content. The operator also extracts all of the tickers that are identified in the communication based on the dictionary. The operator invokes the library if the `executeFlag` of the concept mapper feature is set to `true`.

Input type

FeatureInOut

Output type

FeatureInOut

If the library is invoked and returns Success, the JSON formatted response of the library is populated in the featureOutput element of the concept mapper feature.

Configuration parameters

dictPath—Path of concept mapping library dictionary.
rulePath—Path of concept mapping library rule file.

Document classifier operator

This Java based feature extraction operator invokes the document classifier library to detect if any confidential content is displayed in the communication content. The operator invokes the library if the executeFlag element of the document classifier feature is set to true.

Input type

FeatureInOut

Output type

FeatureInOut

If the library is invoked and returns Success, the JSON formatted response of the library is populated in the featureOutput element of the document classifier feature.

Configuration parameters

dictPath—Path of document classifier library dictionary.
rulePath—Path of document classifier library rule file.

Entity extractor operator

This Java based feature extraction operator invokes the entity extraction library to detect entities, such as organization, tickers, and people, in the communication content. The operator invokes the library if the executeFlag element of the entity extractor feature is set to true.

Input type

FeatureInOut

Output type

FeatureInOut

If the library is invoked and returns Success, the JSON formatted response of the library is populated in the featureOutput element of the entity extractor feature.

Configuration parameters

dictPath—Path of entity extractor library dictionary.
rulePath—Path of entity extractor library rule file.
modelPath—Path of entity extractor library model file.

BehaviorAnomalyRD operator

This Java based risk indicator operator computes risk indicators for anger anomaly, sad anomaly, and negative sentiment anomaly. The operator reads the emotion library response and checks if the anger/sad score exceeds a threshold, and if it does, the operator computes the anomaly score. Similarly, if negative sentiment is reported for the communication, the operator computes the anomaly score. The operator emits risk indicator output if the anomaly score exceeds the anomaly score threshold.

This operator emits the following risk indicators:

- RD1—Anger anomaly
- RD2—Unhappy anomaly
- RD3—Negative sentiment anomaly

Input type

RiskIndicatorIn

Output type

RiskIndicatorOut

A risk indicator is emitted only when the corresponding anomaly score exceeds the anomaly score threshold.

Configuration parameters

angerThreshold—Threshold for the anger score.
sadThreshold—Threshold for the sad score.

selfThreshold—Employee threshold for behavior anomaly.
popThreshold—Threshold for a population, such as all employees, for behavior anomaly.
anomalyThreshold—Threshold for the anomaly score.

CommVolumeAnomalyRD operator

This SPL based risk indicator operator computes risk indicators for inbound anomaly and outbound anomaly. The operator computes the inbound anomaly score of all of the participants in the communication and it computes the outbound anomaly score for the initiator of the communication. The operator emits risk indicator output if the anomaly score exceeds the anomaly score threshold.

This operator emits the following risk indicators:

- RD4—Inbound anomaly
- RD5—Outbound anomaly

Input type

RiskIndicatorIn

Output type

RiskIndicatorOut

A risk indicator is emitted only when the corresponding anomaly score exceeds the anomaly score threshold.

Configuration parameters

selfThreshold—Employee threshold for behavior anomaly.
popThreshold—Threshold for a population, such as all employees, for behavior anomaly.
anomalyThreshold—Threshold for the anomaly score.

CommContentAnomalyRD operator

This Java based risk indicator operator computes risk indicators for communication content anomaly. The operator reads the document classifier library response and checks if the confidential score exceeds a threshold and it computes an anomaly score. Also, top class is reported as confidential. The operator emits risk indicator output if the anomaly score exceeds the anomaly score threshold.

This operator emits the following risk indicators:

- RD6—Confidential anomaly

Input type

RiskIndicatorIn

Output type

RiskIndicatorOut

A risk indicator is emitted only when the corresponding anomaly score exceeds the anomaly score threshold.

Configuration parameters

confidentialThreshold—Threshold for confidential score.
selfThreshold—Employee threshold for confidential anomaly.
popThreshold—Threshold for a population, such as all employees, for behavior anomaly.
anomalyThreshold—Threshold for the anomaly score.

CommunicationAnomalyRD operator

This SPL based risk indicator operator computes risk indicators based on a threshold.

This operator emits the following risk indicators:

- RD7—Unusual mail size. For example, when a mail exceeds a size threshold, emit a score of 1.
- RD8—Unusual number of attachments. For example, when the number of attachments exceed a threshold, emit a score of 1.
- RD9—Unusual communication timings. For example, when mail timing is outside a defined window or mail is sent over the weekend, emit a score of 1.
- RD10—Unusual number of recipients. For example, when the number of mail recipients exceeds a threshold, emit a score of 1.

Input type

RiskIndicatorIn

Output type

RiskIndicatorOut

A risk indicator is emitted only when the corresponding risk indicator score is 1.

Configuration parameters

numberOfRecipients—Threshold number of recipients.

numberOfAttachments—Threshold number of attachments.

windowStartTime—The start time for the window. The format is hh:mm:ss.

windowEndTime—The end time for the window. The format is hh:mm:ss.

totalMailSize—Threshold mail size.

PumpDumpAnomalyRD operator

This Java based risk indicator operator computes risk indicators for pump-and-dump anomaly. The operator reads the concept mapper library response and checks if the recruitVictims or recruitConspirators are reported as `true`. If they are reported as `true`, the following risk indicators are emitted with a score of 1.

- RD22—Recruit victims.
- RD23—Recruit co-conspirators.

Additionally, the operator also computes an anomaly score when recruitVictims or recruitConspirators are reported as `true`. If the anomaly score exceeds the anomaly score threshold, then the operator emits the following risk indicators:

- RD24—Efforts on recruit victims.
- RD25—Efforts on recruit co-conspirators.

Input type

RiskIndicatorIn

Output type

RiskIndicatorOut

Risk indicators for anomaly (RD24 and RD25) are emitted when the corresponding anomaly score exceeds the anomaly score threshold. A risk indicator is emitted when the corresponding risk indicator score is 1.

Configuration parameters

selfThresholdRV—Threshold for an employee for recruit victims.

popThresholdRV—Threshold for a population, such as all employees, for recruit victims anomaly.

selfThresholdRC—Threshold for an employee for recruit conspirators anomaly.

popThresholdRC—Threshold for a population, such as all employees, for recruit conspirators anomaly.

anomalyThreshold—Threshold for the anomaly score.

E-Comm data ingestion

The Surveillance Insight for Financial Services solution expects e-comm data, such as email and chat, in XML format. At least one policy must be defined in the system to be able to process the e-comm data. A policy is a user-defined document that controls the features that need to be extracted.

Once policies are created, e-comm data can be ingested into the solution. Policies can be created and updated by using the REST services. For more information, see [“Policy service APIs”](#) on page 94.

The REST services publish the email and chat data to a Kafka topic. The Streams job that is named ActianceAdaptor reads the data from the Kafka topic. The job parses the data, extracts the necessary elements, and then converts the information into a communication object tuple. This resulting tuple is then further analyzed by the Streams job that is named CommPolicyExecution.

System level policy

System level features are extracted from every communication. The following is an example of a system level policy:

```
{
  "policy": {
    "policyname": "Policy 1",
```

```

    "policycode": "POL1",
    "policytype": "system",
    "policysubcategory": "Sub1",
    "policydescription": "System Policy 1",
    "features": [{
      "name": "emotion"
    }]
  }
}

```

Role level policy

Role level features are extracted based on the initiator party's role and the features that are defined for the role. The following is an example of a role level policy:

```

{
  "policy": {
    "policyname": "Policy 2",
    "policycode": "POL2",
    "policytype": "role",
    "policysubcategory": "Sub2",
    "policydescription": "Role Level Policy",
    "role": [
      "Trader",
      "Banker"
    ],
    "features": [{
      "name": "document classifier"
    }, {
      "name": "concept mapper"
    }, {
      "name": "entity extractor"
    }]
  }
}

```

Sample e-comm email and chat

A sample email xml is available [here](http://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/samplefile/SurveillanceInsightSampleEcommEmail.xml). (www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/samplefile/SurveillanceInsightSampleEcommEmail.xml)

A sample chat xml is available [here](http://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/samplefile/SurveillanceInsightSampleEcommChat.xml). (www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/samplefile/SurveillanceInsightSampleEcommChat.xml)

Data flow for e-comm processing

The solution provides features to process electronic communication data such as email and chat transcripts.

The following diagram shows the end-to-end data flow for e-comm processing:

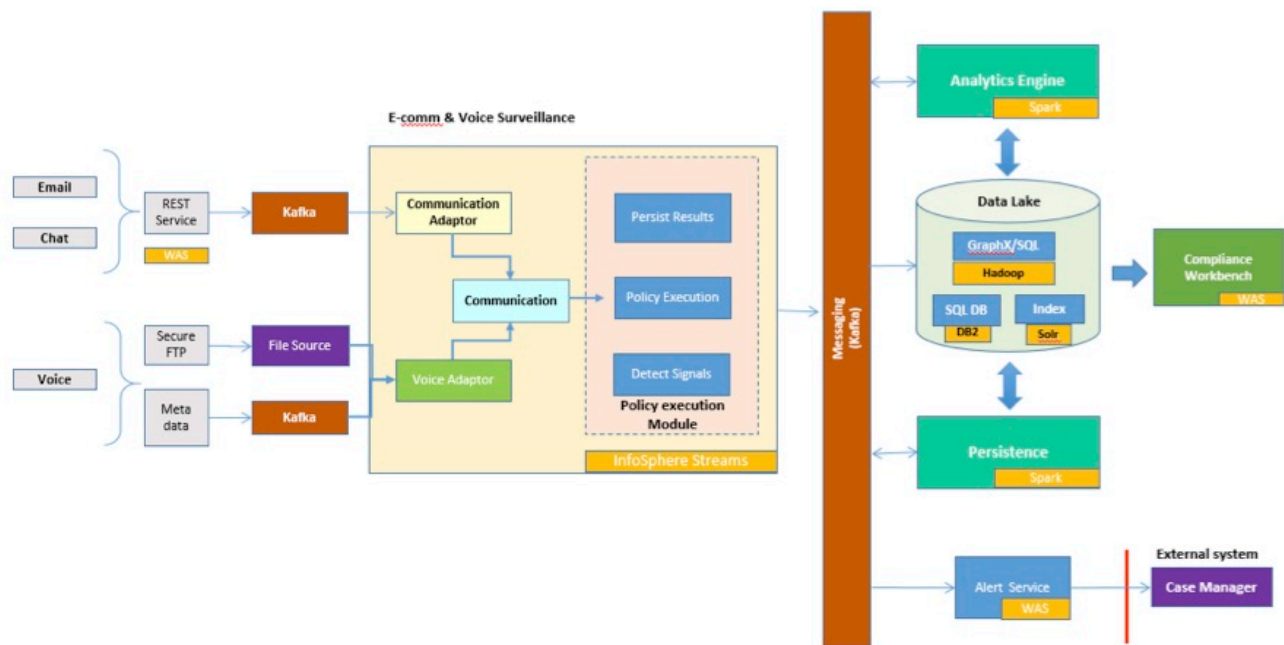


Figure 24: e-comm surveillance data processing

- e-comm data is fed into the Kafka topic and monitored by the ActianceAdaptor Streams job.
- The ActianceAdaptor stream converts the xml message into a communication tuple.
- The CommPolicyExecution Streams job processes the communication tuple against the policies that are applicable for the communication and then creates a JSON message that contains the communication data along with the extracted features and risk indicator details.
- The JSON message is published to the Kafka topic. And the topic is consumed by the ECommEvidence Spark job.
- The ECommEvidence Spark job saves the communication data and the extracted features and risk indicators to the database. Also, Solr runs the inference engine to check for alertable conditions. If so, an alert is created in the system.
- Alerts are created or updated in the database based on the outcome of the inference engine. This is done through the Create Alert REST service.

The following diagram shows the end-to-end data flow:

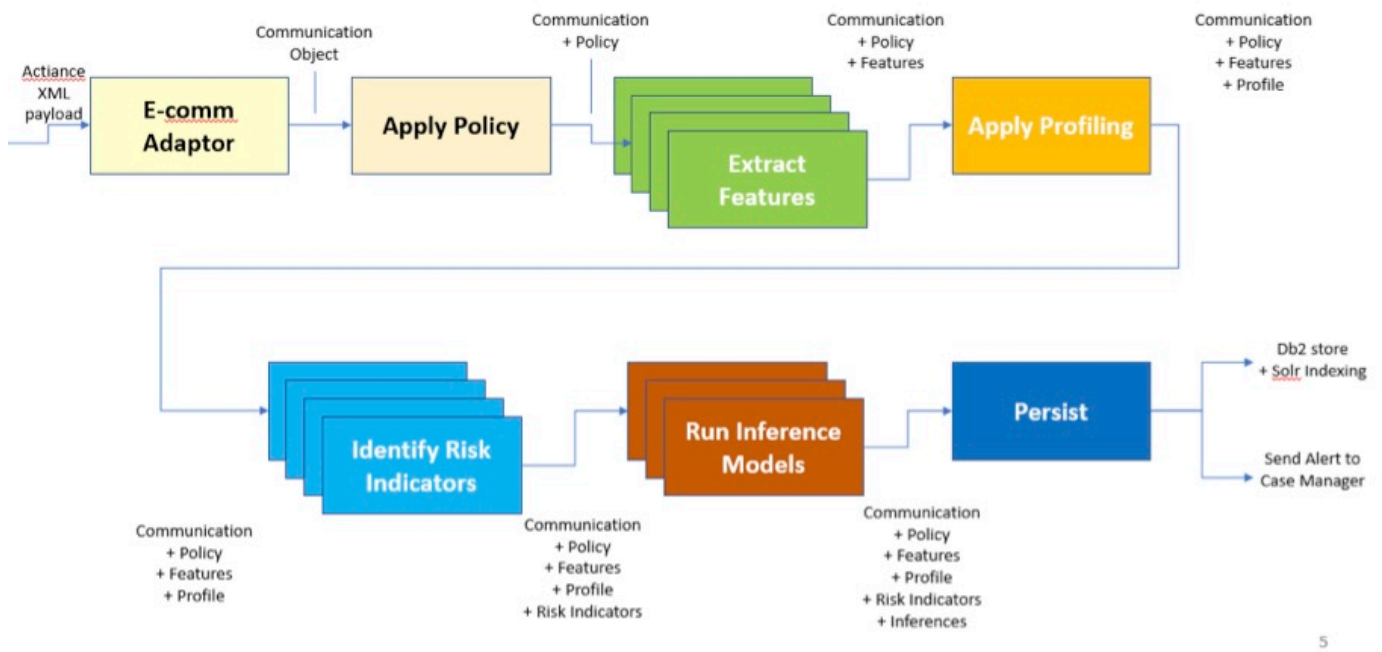


Figure 25: e-comm surveillance data

E-Comm use case

E-Comm data, such as email and chat transcripts, are published as XML files to Surveillance Insight for Financial Services.

Surveillance Insights reads the XML files and parses them by using an IBM InfoSphere Streams job named ActianceAdaptor. ActianceAdaptor converts the XML content into a communication tuple under the CommData schema. The ActianceAdaptor reads data from a Kafka topic, parses it, and then converts it to the CommData schema. The adaptor also submits the data for further processing to other jobs, such as CommPolicyExecution.

The ActianceAdaptor job is shown following diagram.

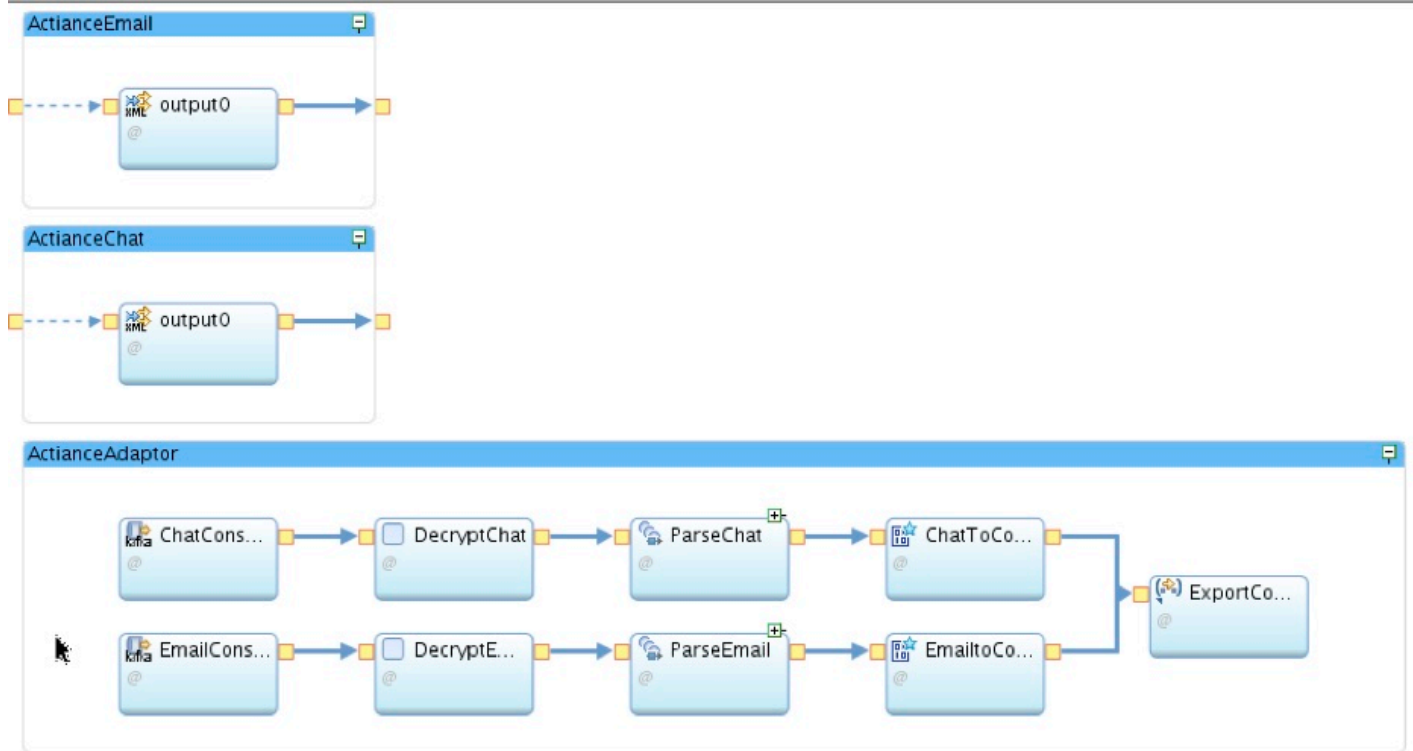


Figure 26: ActianceAdaptor job

CommPolicyExecution job

The CommData is processed further by the CommPolicyExecution job. The CommPolicyExecution job processes data from all channels, such as email, chat, and voice. The job does following activities:

1. The CommPolicyExecution job loads the following data into Streams.
 - All Parties and their contact points that are registered in the Surveillance Insight database.
 - Reference profiles for all parties and risk indicators for the latest date for which the profile (that is, MEAN) is populated, and the population profile for all risk indicators for the latest date that is available in the Surveillance Insight database.
 - All policies that are registered in the Surveillance Insight database.
2. The CommPolicyExecution job is triggered when the following types of data are sent:
 - CommData—when the input data is of CommData type, the job processes the data further in the next step.
 - Policy—when a new policy is created or an existing policy is updated, a Kafka message is sent to the sifs.policy.in topic, and the job reads the policy details and refreshes the Streams memory with the updated policy details.
3. Then, the job reads the initiator contact details and, if no initiator details are found, further processing is skipped and a JSON formatted message is published to the sifs.ecomm.out topic for downstream processing.
4. If initiator details are found in Surveillance Insight, the job retrieves all of the associated policies for the initiator contact point. If no policy is found, the job logs a message and no further processing is done. The JSON formatted message is published to the sifs.ecomm.out topic for downstream processing.
5. If any policies are found for the initiator, all of the policies are parsed and a consolidated feature map is created. The feature map is applied for the incoming communication and a new tuple of type FeatureInOut is created.
6. The FeatureInOut tuple is further processed by other feature extraction operators in parallel and the consolidated feature map with results from all the feature extraction operators is combined for further processing.
7. The job reads the initiator profile and appends to the incoming FeatureInOut tuple to create a new tuple of type RiskIndicatorIn. The profile of the user is refreshed, depending on the configured frequency. The frequency is configured when the CommPolicyExecution job is submitted.

8. The RiskIndicatorIn is further processed by other risk indicator operators in parallel and the consolidated risk indicator map with the risk indicators that are identified for the communications are combined and further processed. The consolidated result is returned by a new tuple of type RiskIndicatorOut.
9. The RiskIndicatorOut tuple is published to the sifs.ecomm.out topic for downstream processing.

The CommPolicyExecution job is as shown in the following diagram:

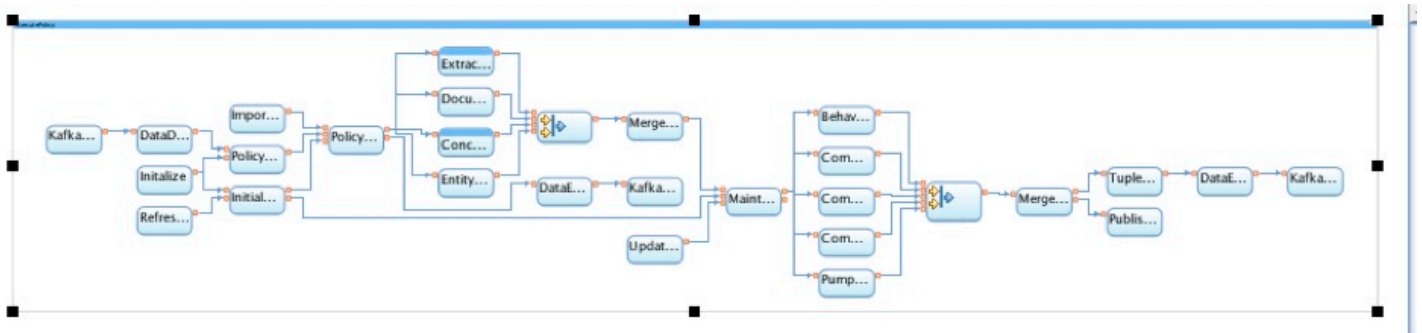


Figure 27: CommPolicyExecution job

ECommEvidence job

This job reads messages from sifs.ecomm.out Kafka topic and persists the communication data in the Surveillance Insight database and in Solr. The e-comm data is further analyzed by the inference engine against the defined risk models. The Party Behavior Risk model is provided with the solution. The following diagram explains the tables that are populated when e-comm data is published to the sifs.ecomm.out topic.

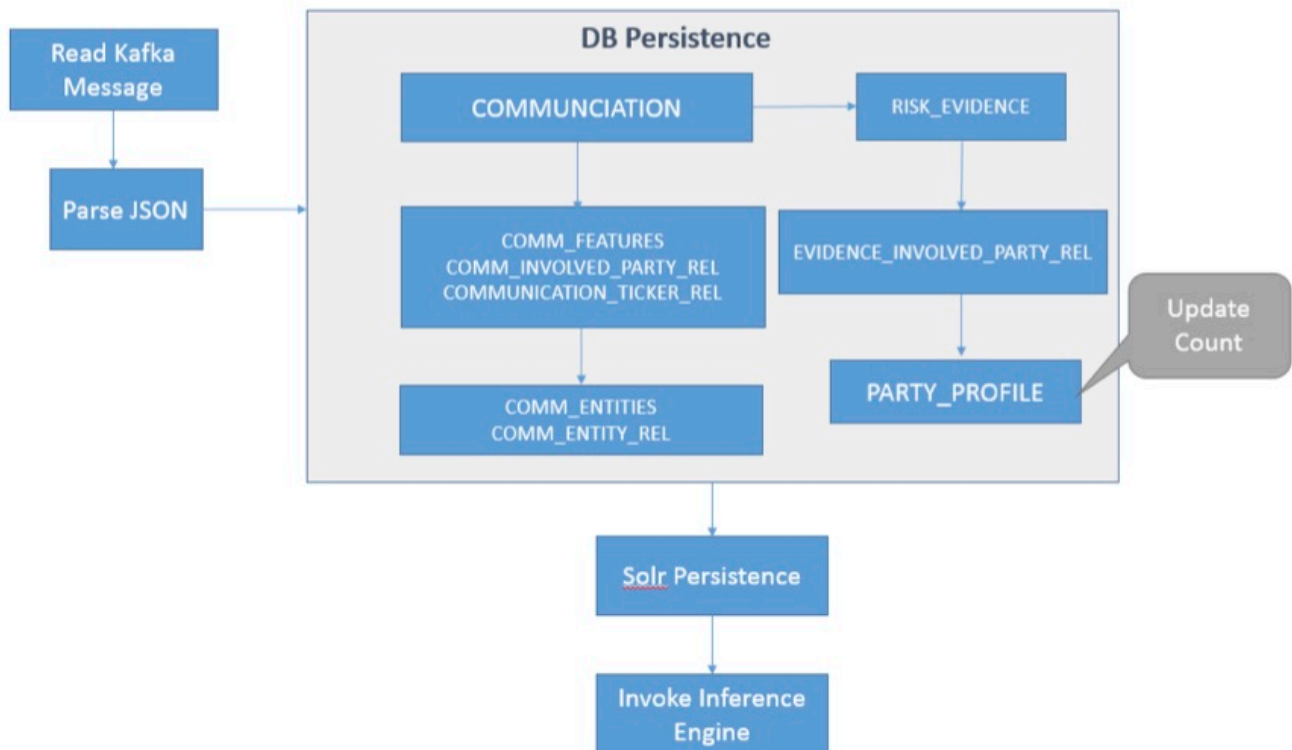


Figure 28: ECommEvidence job

BehaviorProfile job

This job computes the behavior profile for all parties for a date. The job requires the date to be set in the `sifs.spark.properties` file as the `behaviorProfileDate` property. The `behaviorProfileDate` is the date for which the MEAN and STD must be calculated.

This job inserts data in `PARTY_BEHAVIOR_PROFILE` for the `behaviorProfileDate`. The job must be executed after the date is set. Surveillance Insight expects that this job is run for a date only one time. If the same job is run for the same date more than one time, an exception is logged by the job. Only INSERT functions are supported. UPDATE functions are not.

ProfileAggregator job

This job computes the profile for all parties for a date. The job expects the date to be set in the `sifs.spark.properties` file in the following properties:

- `profileDate`—the date for which the MEAN and STD must be calculated.
- `window`—the number of days for which the MEAN and STD need to be calculated.

This job updates the MEAN and STD values in the `PARTY_PROFILE` for the profile date. It also inserts the MEAN, STD, and COUNT in the `ENTERPRISE_POFILE` for the profile date. Surveillance Insight expects that the job is run for a date only one time. If the same job is run for the same date more than one time, an exception is logged by the job. Only INSERT functions are supported. UPDATE functions are not.

E-Comm Toolkit data schemas

The Surveillance Insight for Financial Services database schema holds data for the major entities of the business domain.

The major entities are:

- Party
For more information, see [“Party view” on page 46](#).
- Communication
For more information, see [“Communication view” on page 48](#).
- Alert
For more information, see [“Alert view” on page 50](#).
- Trade
For more information, see [“Trade view” on page 52](#).

The following diagram shows the components that update the key sets of tables.

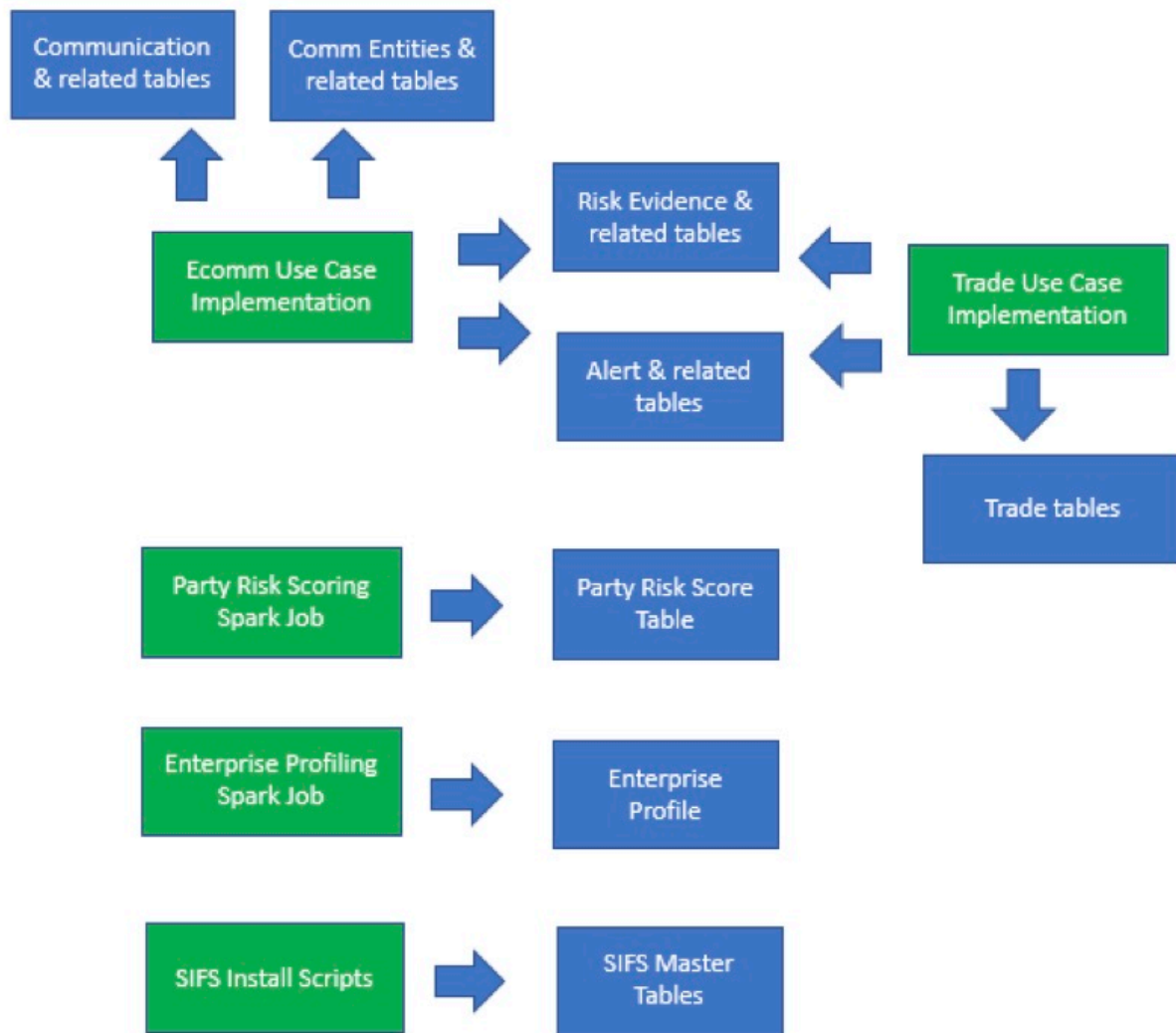


Figure 29: Components that update key tables

Party view

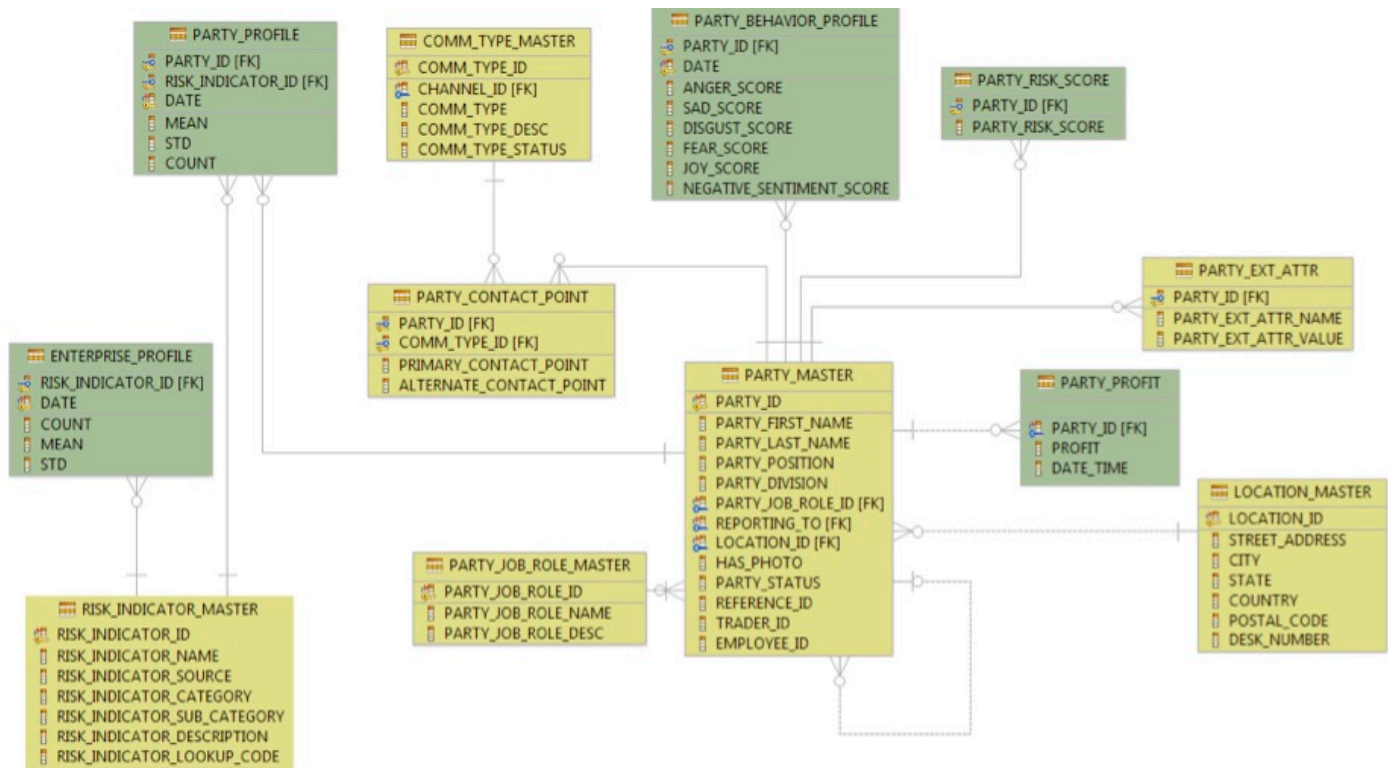


Figure 30: Party view schema

Table name	Description	Populated by
Party_Master	Stores basic party details. Reference ID refers to id in the core systems. TraderId contains mapping to the id that the party uses for trading.	Populated during initial data load and periodically kept in sync with the customer's core master data system.
Party Risk Score	Stores overall risk score for the party based on the party's current and past alert history.	Scheduled Surveillance Insight job that runs daily.
Party Behavior Profile	Stores date-wise scores for various behavior parameters of the party.	Surveillance Insight Ecomm Job daily.
Party Profile	Stores date wise, risk indicator wise statistics for every party.	Surveillance Insight Ecomm Job that updates count field for every communication that is analyzed. Updates the other fields on a configurable frequency.
Enterprise Profile	This table maintains the Mean and Std Deviation for each date and Risk Indicator combination. The mean and standard deviation is for the population. For example, the values are computed using data for all parties.	This table is populated by a Spark job that is based on the frequency at which the Spark job is run. The job reads the date parameter and, for that date, populates the Enterprise profile table. The solution expects to populate the data in the Enterprise profile only once for a specific date.

Table name	Description	Populated by
Party Contact Point	Contains email, voice, and chat contact information for each party.	Populated during the initial data load and periodically kept in sync with the customer's core master data system.
Party Profit	Profit made by the party from trades.	Populated by the spoofing use case implementation.
Party Job Role Master	Master table for party job roles.	Populated during the initial data load and periodically kept in sync with the customer's core master data system.
Comm Type Master	Master table for communication types such email, voice, and chat.	Populated during the initial data load and periodically kept in sync with the customer's core master data system.
Location Master	Master table with location details of the parties.	Populated during the initial data load and periodically kept in sync with the customer's core master data system.
Party_Ext Attr	Table to allow extension of party attributes that are not already available in the schema.	Populated during implementation. Not used by the provided use cases.

Communication view

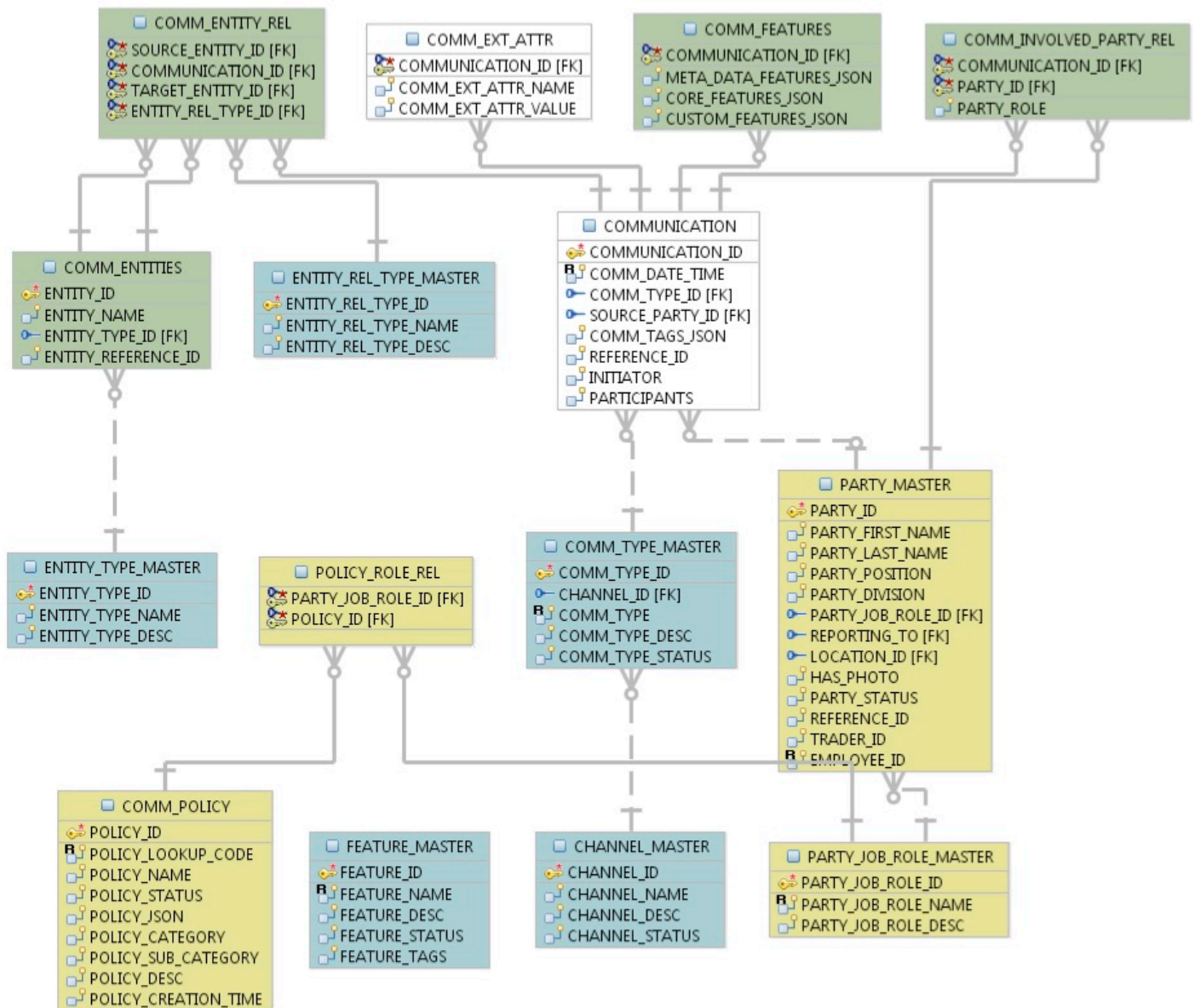


Figure 31: Communication view schema

Table name	Description	Populated by
Communication	Core table that stores extracted metadata from electronic communications (e-comm). It does not store the actual content of the email or voice communication. It stores data such as the initiator, participants, and associated tags.	This table is populated by the Surveillance Insight e-comm stream that analyzes the e-comm data when each communication comes in.
Comm Involved Party Rel	Stores parties that are involved in a communication.	This table is populated by the Surveillance Insight e-comm stream that analyzes e-comm data when each communication comes in.

Table name	Description	Populated by
Comm Entities	Stores entities that are extracted from electronic communications.	Populated by the Surveillance Insight e-comm components during e-comm analysis.
Comm Entities Rel	Stores relationships between entities that are extracted from the electronic communications.	Populated by the Surveillance Insight e-comm components during e-comm analysis.
Comm Policy	This table maintains the policy details registered in Surveillance Insight. The table has data for both system and role level policy.	Populated through the Policy REST service. The service supports create, update, activate, and deactivate features.
Policy Role Rel	This table maintains the policy to role mapping. For role level policies, the relationship for policy and job role is stored in this table.	Populated when the policy is created in the system by using the REST service. Updates to this table are not supported. It is recommended to create a new policy if there any changes in role.
Feature Master	This table contains a master list of all of the features that are supported by Surveillance Insight for Financial Services.	Master table. Populated during Surveillance Insight product setup.
Comm Feature	This table contains the feature JSON for each communication that is processed by Surveillance Insight for Financial Services. The table has a column (CORE_FEATURES_JSON) that contains the JSON for all of the features in Feature master. For the metadata, the JSON is stored in the META_DATA_FEATURES_JSON column. The table also provides a provision to store custom feature values in the CUSTOM_FEATURES_JSON column.	This table is populated for every communication that is processed by the Surveillance Insight for Financial Services.
Comm Type Master	Master table that stores the different communication types such as voice, email, and chat.	Populated with the supported communication types during product installation.
Channel Master	Master table that stores the different communication channels.	Populated with the supported channel types during product installation. The channels are e-comm and voice.
Entity Type Master	Master table for the type of entities that are extracted from the electronic communications.	Populated with the supported types during product installation. The types are people, organization, and ticker.
Entity Rel Type Master	Master table for types of relationships that are possible between entities that are extracted from the electronic communications.	Populated with the supported types during product installation. The types are Mentions and Works For.
Comm Ext Attr	Extension table that is used to store additional communication attributes during customer implementation.	

Alert view

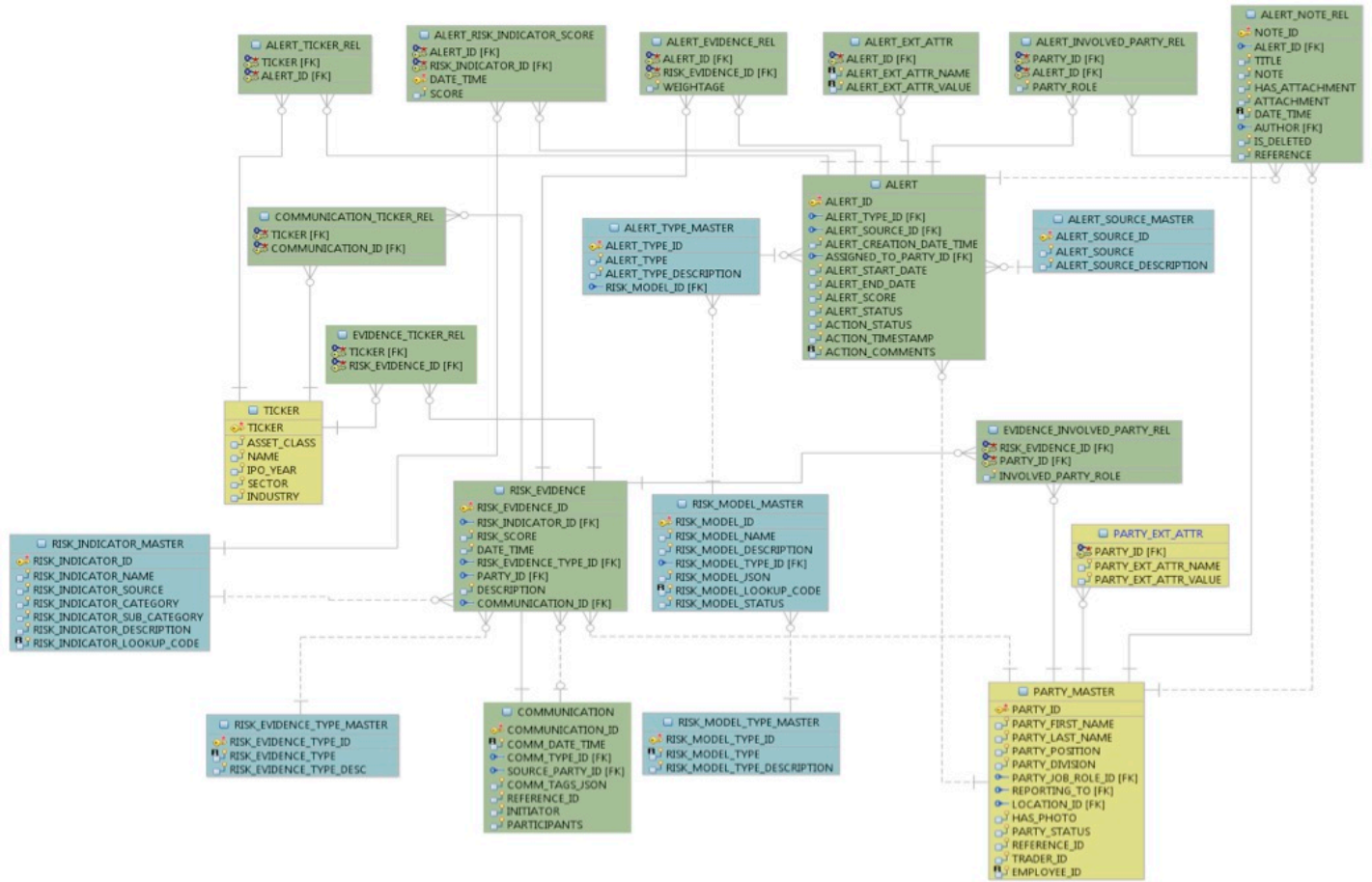


Figure 32: Alert view schema

Table name	Description	Populated by
Alert	Core table that stores the alert data.	This table is populated by any use case that creates an alert. The createAlert REST service must be used to populate this table.
Risk Evidence	Core table that stores each of the risk evidences that are identified during the data analysis.	This table is populated by any use case that creates risk evidences. The createEvidence REST service can be used to populate this table.
Alert Evidence Rel	Links an alert to multiple evidences and evidences to alerts.	This table is populated by any use case that creates an alert. The createAlert REST service must be used to populate this table.
Alert Involved Party Rel	Links the parties who are involved in an alert with the alert itself.	This table is populated by any use case that creates an alert. The createAlert REST service must be used to populate this table.
Alert Risk Indicator Score	Identifies the risk indicators and corresponding scores that are associated with an alert.	This table is populated by any use case that creates an alert. The createAlert REST service must be used to populate this table.

Table name	Description	Populated by
Alert Ticker Rel	Links the tickers that are associated with an alert to the alert itself.	This table is populated by any use case that creates an alert. The createAlert REST service must be used to populate this table.
Alert Note Rel	Stores all the notes that are created by the case investigators for the alerts.	The Surveillance Insight note service populates this table when it is triggered from the product user interface.
Evidence Involved Party Rel	Links the parties that are involved in a risk evidence to the evidence itself.	This table is populated by any use case that creates risk evidences. The createEvidence REST service can be used to populate this table.
Evidence Ticker Rel	Links any tickers that are associated with an evidence to the evidences itself.	This table is populated by any use case that creates risk evidences. The createEvidence REST service can be used to populate this table.
Alert Type Master	Master table for the various types of alerts that can be created by the use cases.	Populated with certain alert types during product installation. More types can be created when you create new use cases.
Alert Source Master	Master table for source systems that can generate alerts.	Populated with one alert source for Surveillance Insight during product installation. More sources can be created, depending on the requirements.
Risk Indicator Master	Master table for risk indicators that are generated by the use cases.	Populated with certain indicators for e-comm and trade risk during product installation. More indicators can be created, depending on the requirements.
Risk Evidence Type Master	Master table for evidence types, such as trade, email, and voice.	Populated with a certain type during product installation. More types can be created, depending on the requirements.
Risk Model Master	Master table for the risk models that are used for generating the reasoning graph.	Populated during the product installation with following models: pump-and-dump, spoofing, and party risk behavior. More models can be populated, depending on the requirements of new use cases.
Risk Model Type Master	Master table for the types of risk models that are supported.	Populated during the product installation with rule and Bayesian network types.
Comm Ticker Rel	Links the tickers that are found in electronic communications to the communication itself.	Populated by the e-comm component for each electronic communication that is analyzed.

Table name	Description	Populated by
Alert Ext Attr	This table allows for the extension of alert attributes.	Not used by Surveillance Insight for Financial Services. This table is meant for customer implementations, if required.

Trade view

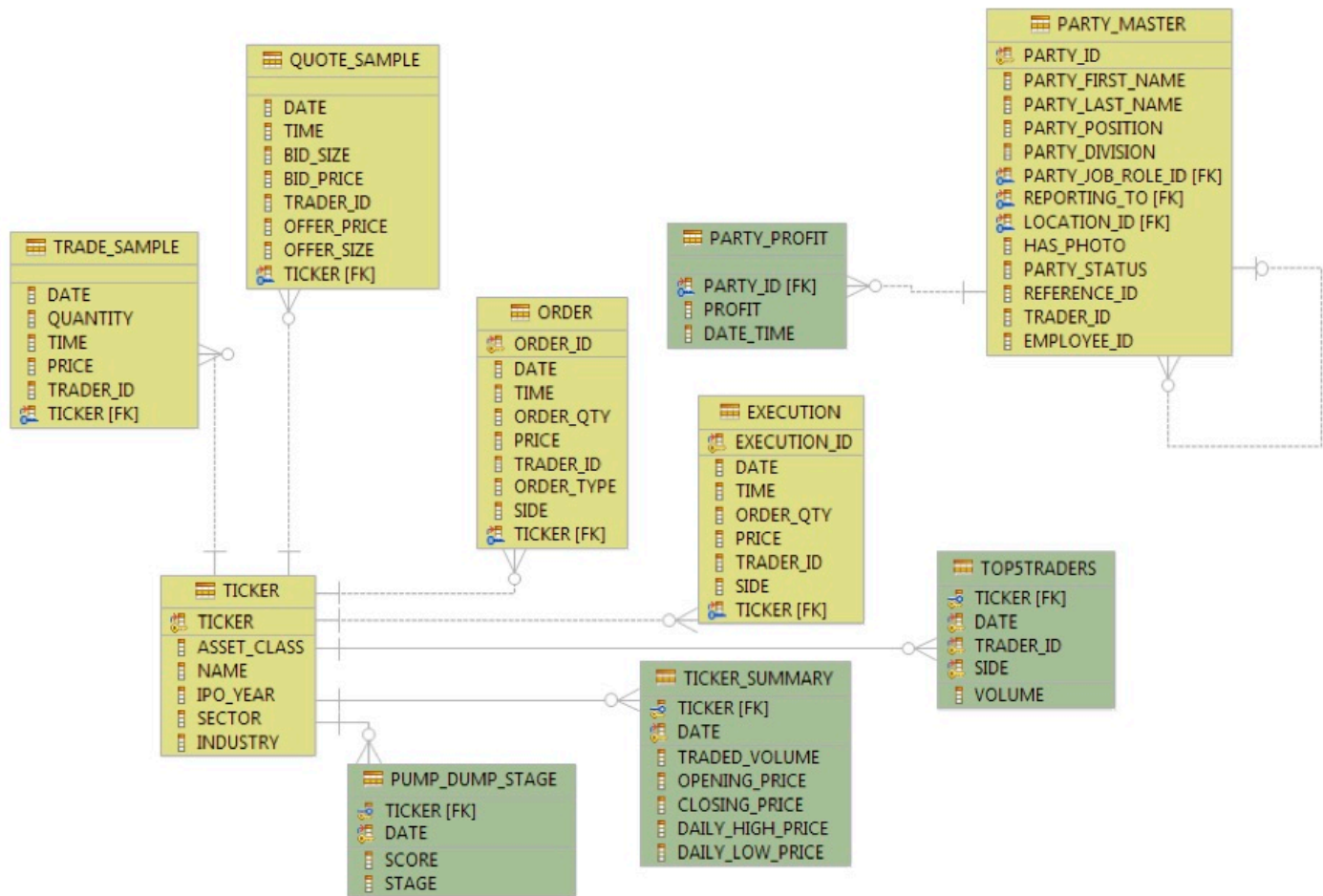


Figure 33: Trade view schema

Table name	Description	Populated by
Ticker	Master table that stores basic ticker information.	This table is populated during the initial data load and whenever new tickers are found in the trade data.

Table name	Description	Populated by
Trade Sample	This table contains samples of trade data from the market. The trades that go into this table depend on the specific use case. The trades are primarily evidences of some trade risk that is detected. Typically, these trades are sampled from the market data that is stored in Hadoop. They are stored here for easy access by the user interface layer.	Currently, the spoofing and pump-and-dump use cases populate this table whenever a spoofing alert is identified.
Quote Sample	This table contains samples of quote data for certain durations of time. The quotes that go into this table depend on the specific use case. The quotes are evidences of some kind of risk that is identified by the specific use case. These quotes are sampled from the market data and stored in Hadoop. They are stored in this table for easy access by the user interface layer.	Currently, the spoofing use case populates this table whenever a spoofing alert is created. The sampled quote is the max (bid price) and min (offer price) for every time second.
Order	This table contains orders that need to be displayed as evidence for some alert in the front end. The contents are copied from the market data in Hadoop. The specific dates for which the data is populated depends on the alert.	Currently, the spoofing use case populates this table whenever a spoofing alert is identified.
Execution	This table contains orders that need to be displayed as evidence for some alert in the front end. The contents are copied from the market data in Hadoop. The specific dates for which the data is populated depends on the alert.	Currently, the spoofing use case populates this table whenever a spoofing alert is identified.
Pump Dump Stage	This table contains the pump-and-dump stage for each ticker that shows pump or dump evidence.	This table is populated by the pump-and-dump use case implementation.
Trade Summary	This table contains the ticker summary for each date for tickers that show pump or dump evidence.	This table is populated by the pump-and-dump use case implementation.
Top5Traders	This table contains the top five traders for buy and sell sides for each ticker that shows pump or dump evidence. This table is populated daily.	This table is populated by the pump-and-dump use case implementation.

Extending E-Comm Surveillance

Solution developers can use the solution's architecture to develop new e-comm surveillance use cases.

The following example adds a feature to the solution to detect offline communications.

Procedure

1. Create a library that can detect offline communications.

2. Create a Feature Extraction Operator. Ensure that the input and output for the Feature Extraction Operator have FeatureInOut as the type.
3. Create a Risk Indicator Operator to detect whether the communication detects any offline occurrences. If it does, mark those occurrences by emitting a risk indicator output and associating all evidences with it.
4. Build a risk model that uses the evidences to determine whether an alert needs to be generated. See the section on the Inference Engine for more information on how to develop a risk model.
5. The type of alert that are generated by the use case must be identified.
 - a) Add a feature in the FEATURE_MASTER table.
 - b) Add new risk indicators in the RISK_INDICATOR_MASTER table
 - c) Populate the risk model in the RISK_MODEL_MASTER table.
 - d) Populate the alert type in the ALERT_TYPE_MASTER table.
6. Run the new operators through the Streams job that is named CommPolicyExecution. You must add the operator to the existing spl file and modify the configuration files for the new library.
7. The existing Spark job that is named ECommEvidence persists the new operators in the database and in Solr if you add the operators to the Streams job.
8. Create a new Spark job to run the new inference model if the risk indicators need a new risk model.
9. After the new inference model is created, configure the ECommEvidence job to invoke the new model so that every communication can be analyzed through that new model.

Chapter 5. Voice surveillance

The voice component processes voice data files either in WAV or PCAP formats into text. The text from the voice data files is then evaluated against various features and different risk indicators are calculated. The risk indicators are then analyzed by the inference engine to detect alarming conditions and generate alerts if needed.

Data ingestion

IBM Surveillance Insight for Financial Services processes voice data in the following formats:

- WAV file in uncompressed PCM, 16-bit little endian, 8 kHz sampling, and mono formats
- PCAP files and direct network port PCAP

At least one policy must be defined in the system to be able to process voice data.

To create a voice WAV sample, you can use any voice recording software that allows you to record in the accepted WAV formats.

Metadata captures voice as a CSV file by using the following schema:

```
#initiator-partyid,initiator_phone_num,initiator_device_id,participants_partyid,participants_phone_num,participants_device_id,voice_file_name,date,callstarttime,callendtime,analysis_policy_id,global_comm_id
```

A voice metadata script is provided to help post encrypted messages to Kafka. Use the following command to run the script.

```
./processvoice.sh voicemetadata.csv
```

The following diagram shows the data flow for voice surveillance.

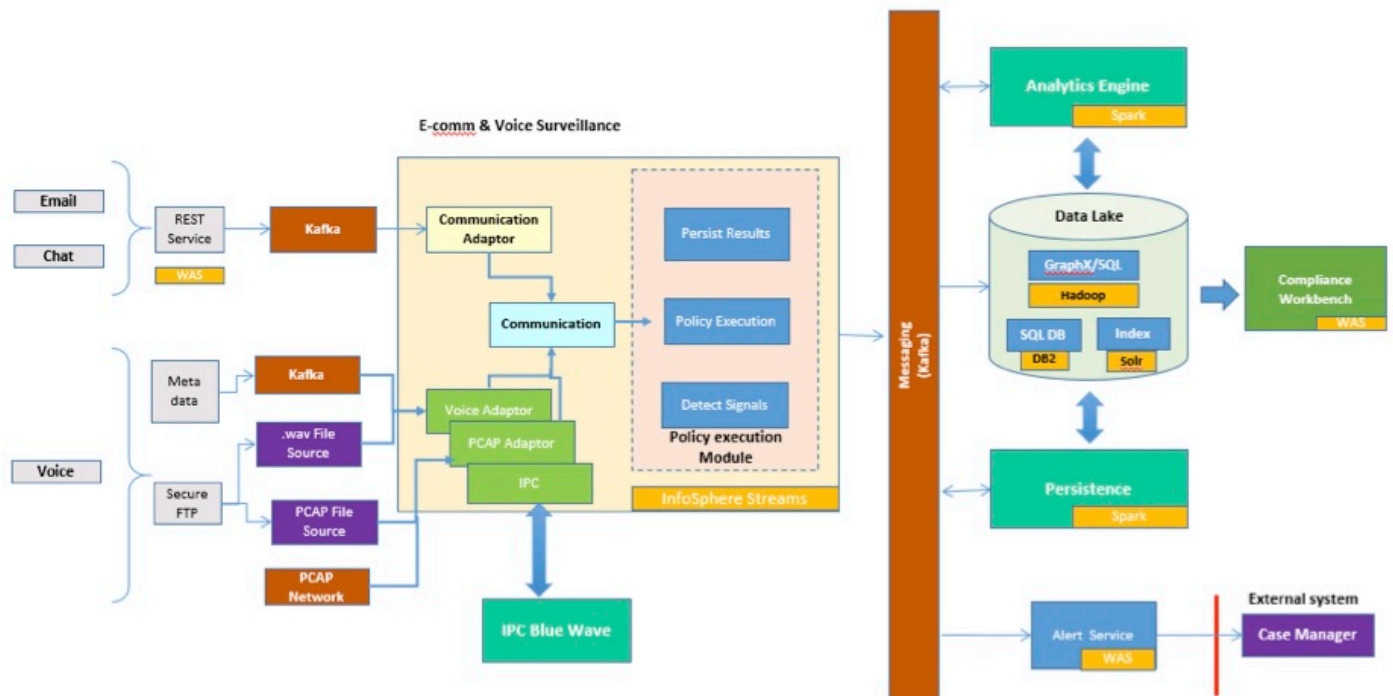


Figure 34: Data flow for voice surveillance

1. Voice data is loaded in. For WAV files, the metadata is fed into the Kafka topic.
2. The Voice Adaptor convert the content into a communication tuple.

3. The CommPolicyExecution streams job processes the communication tuple against the applicable policies for the communication and creates a JSON message that contains the communication data and the extracted features and risk indicator details.
4. The JSON message is published to the Kafka topic that is consumed by the ECommEvidence Spark job.
5. The ECommEvidence Spark job persists the communication data along with the extracted features and risk indicators to the database and to Solr. The job also invokes the inference engine to determine if there are any alertable conditions. If so an alert is created in the system.
6. Alerts are created or updated in the database based on the outcome of the inference engine. This is done through the Create Alert REST service.

The following diagram shows how WAV files are processed.

■ WAV Processing Flow

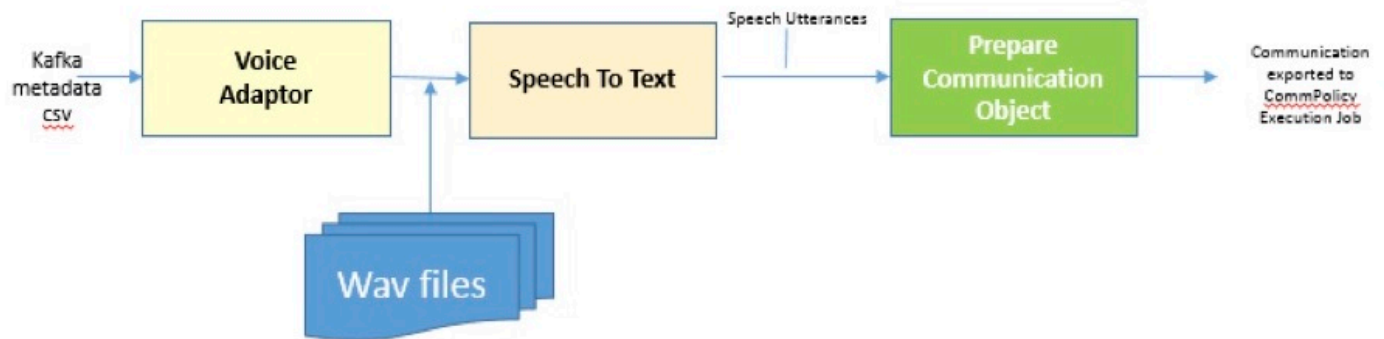


Figure 35: WAV file processing

The following diagram shows how PCAP files are processed.

■ PCAP and IPC Processing Flow

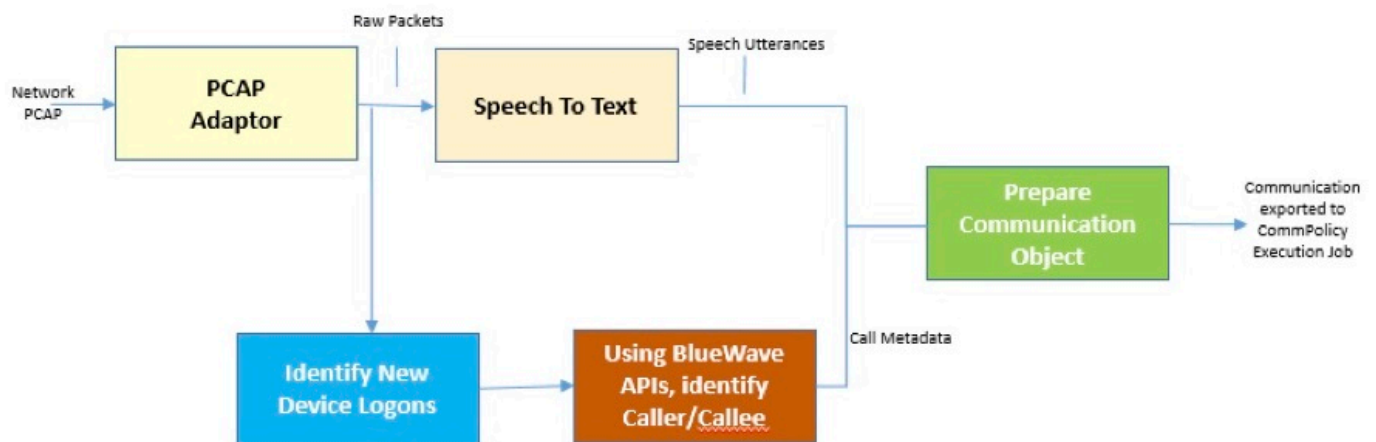


Figure 36: PCAP file processing

Voice Surveillance metadata schema for the WAV Adaptor

The following table shows the metadata schema for the WAV adaptor.

Table 10: Metadata schema for the WAV adaptor			
S.no	Field Name	Field Type	Description
1	partyid	String	Call Initiator's Party id
2	initiator_phone_num	String	Phone number of call Initiator
3	initiator_device_id	String	Device id from which the call was initiated
4	participants_partyid	String	Partyid of receiving participants. Multiple values are separated by semi-colons.
5	participants_phone_num	String	Phone number of receiving participants. Multiple values are separated by semi-colons.
6	participants_device_id	String	Device id of receiving participants. Multiple values are separated by semi-colons.
7	voice_file_name	String	Audio file name that needs to be processed.
8	date	String	Date the call was recorded in YYYY-MM-DD format.
9	callstarttime	String	Call start time in hh:mm:ss format.
10	callendtime	String	Call end time of call in hh:mm:ss format.
11	analysis_policy_id	String	Policy ID that should be applied while analyzing this audio. This is optional.
12	global_comm_id	String	Unique global communication id attached to this audio.

WAV format processing

Voice communications in WAV format are processed through different adaptors, Kafka processes, and Streams jobs.

IBM Surveillance Insight for Financial Services processes WAV files based on the metadata trigger that is received through the pre-defined Kafka topics. The WAV Adaptor reads the data from the Kafka topic, decrypts the Kafka

message, parses it, and fetches the voice WAV file name. The voice WAV file name is passed to the SpeechToText (S2T) toolkit operator for translation. All of the utterances from S2T are aggregated. The aggregated text is then converted to the CommData schema and submitted to the CommPolicyExecution job by using the Export operator for further processing.

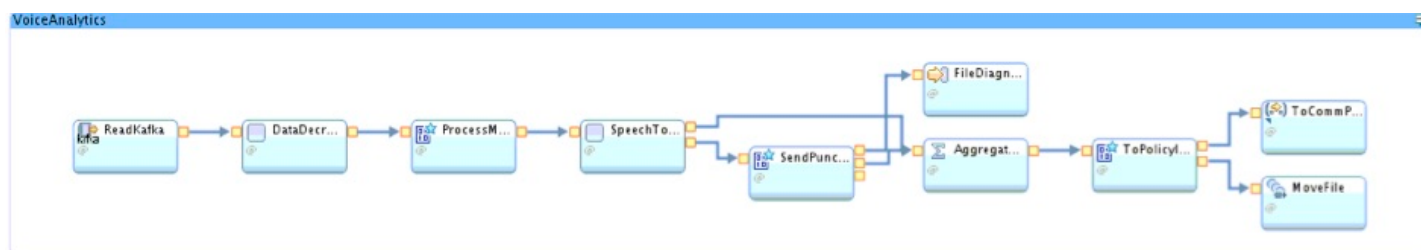


Figure 37: WAV

PCAP format processing

Processing of PCAP format files involves PCAP speech extraction and IPC-based call metadata extraction.

PCAP Adaptor job

The PCAP Adaptor job parses PCAP data either from a PCAP file or from a network port. The raw packet data is also exported to the IPC job. Packets are filtered based on IPs and Subnets. The filtered RTP packets are processed and all of the PCAP binary data is collected for a call. The aggregated blob RTP data is then written out to a temporary location. Certain call attributes, such as the callid, channel_id, source, and destination port, are exported to the CallProgressEvent job.

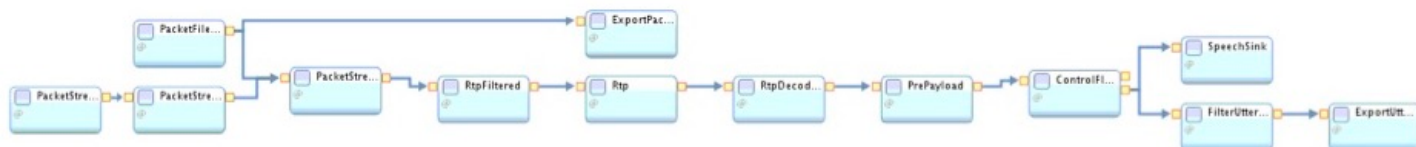


Figure 38: PCAP Adaptor job

Watson job

This job contains the SpeechToText operators for processing PCAP audio data. The temporary blob data files that are written by the PCAP Adaptor are run through S2T operator to translate the speech data into text. The converted text is buffered until the call metadata tuple arrives. The call metadata is correlated with the call's converted text and a CommData object is created and submitted to the CommPolicyExecution job by using the Export operator for further processing.

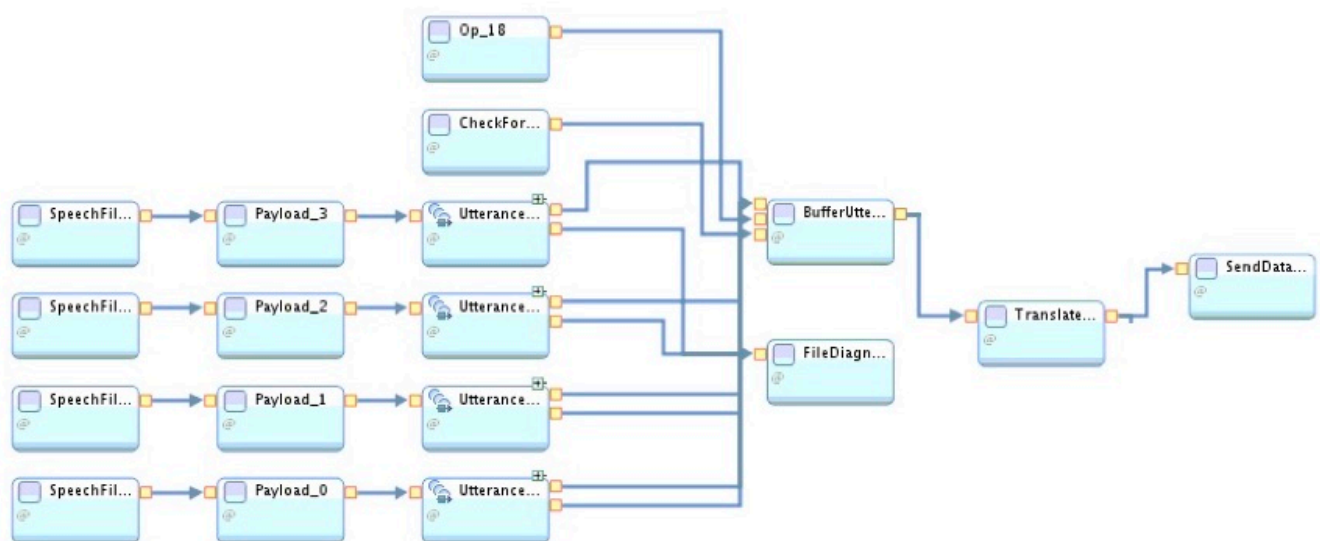


Figure 39: Watson job

IPC metadata job

The IPC metadata extraction jobs consists of three SPLs. CommData is processed further by the CommPolicyExecution job. IPC job is linked to the PCAP Adaptor and reads the raw socket data. It identifies SIP Invite messages for the new users who login to their turret devices. It parses the XML data packets to fetch the DeviceID and sessionID that correspond to handsets and stores them in an internal monitored list. This is done to avoid monitoring audio data from speaker ports. After the SIP ACK messages are received, it verifies that the deviceid from the ACK is present in the monitored list. Then, it emits the DeviceID and the destination port.



Figure 40: IPC metadata job

ConfigureCallProgressEvent job

The DeviceID that is received from the IPC metadata job is further processed by the ConfigureCallProgressEvent job to create appropriate monitors in the BlueWave server to monitor calls that are made by a user through handset channels of turret. For the incoming DeviceID, the LogonSession details are received from the BlueWave (BW) server. From the LogonSession response XML, certain attributes about the current usage of the turret device, such as IP address, zoneid, zonename, loginname, firstname, lastname, and emailid, are stored. When the job starts, the logonsession details for all of the users who are currently logged in to the BW system are fetched and stored.

Any access to BW needs an authentication token. After an authentication token is created, it is refreshed at regular intervals. Call monitors are created in BW for all of the logonsessions that are fetched at job startup and the new logonsessions. While invoking BW REST API for creating Call monitors, the monitor type is set to CallProgress and a callback URL is provided to allow BW APIs to send the callprogress events to. When a call ends, a tuple that contains the source and destination port and a callid is received by this job from the PCAP main job.

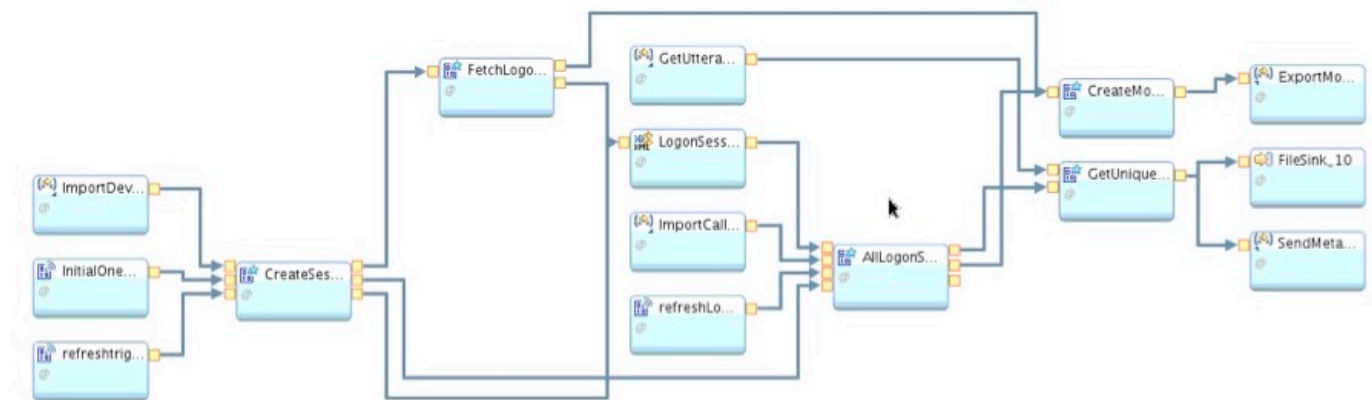


Figure 41: ConfigureCallProgressEvent job

CallEvents job

This job acts as an HTTP receiver of all call events that are sent by BW as a result of setting up of monitors for the turret devices. Typically, event XMLs are emitted for the following event types:

- OriginatedEvent – When a user lifts the handset to dial a number to make calls.
- ConfirmedEvent – When a call is established between a caller and a callee.
- ReleasedEvent - When a call ends and the headset is replaced back.

CallEvents hosts an HTTP port that receives all call progress events that are pushed by BW. On arrival of OriginatedEvent, the callID and the call start time is logged. On arrival of ConfirmedEvent, the callee and caller login names are registered against the callID. On arrival of ReleasedEvent, the call end time is registered against the callID. After the ReleasedEvent arrives, a callmetadata tuple that contains the callID, login names for caller and callee, and the call start and end timestamps are sent to the ConfigureCallprogress job.

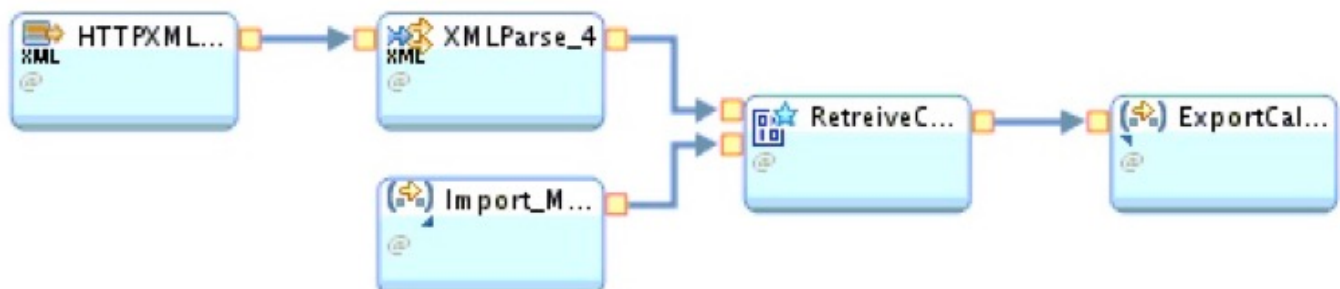


Figure 42: CallEvents job

Chapter 6. NLP libraries

The solution offers prebuilt custom libraries for some of the Natural Language Processing (NLP) capabilities

The following pre-built libraries are provided:

- Emotion Detection library
- Concept Mapper library
- Document Classifier library

The solution uses Open source frameworks / Libraries such as [Apache UIMA](#) (Unstructured Information Management Application) and [MALLET](#) (MACHINE Learning for Language Toolkit).

Note: The libraries come with dictionaries and rules that can be customized.

Emotion Detection library

The Emotion Detection library uses Apache UIMA Ruta (RULE based Text Annotation) and a custom scoring model to detect emotions and sentiment in unstructured data, such as text from emails, instant messages, and voice transcripts.

The library detects the following emotions from the text:

- Anger
- Disgust
- Joy
- Sadness
- Fear

It assigns a score from 0-1 for each emotion. A higher value indicates a higher level of the emotion in the content. For example, an Anger score of 0.8 indicates that the anger is likely to be present in the text. A score of 0.5 or less indicates that anger is less likely to be present.

The library also detects the sentiment and indicates it as positive, negative, or neutral with a score of 0-1. For example, a positive sentiment score is 0.8 indicates that positive sentiment is likely expressed in the text. A score of 0.5 or less indicates that positive sentiment is less likely expressed in the text. The sentiment score is derived from the emotions present in the text.

How the library works

The solution uses dictionaries of emotions and rules to detect the emotions in text and a scoring model to score the emotions.

The dictionaries are contained in the `anger_dict.txt`, `disgust_dict.txt`, `fear_dict.txt`, `joy_dict.txt`, and `sad_dict.txt` files. Each dictionary is a collection of words that represent emotion in the text.

The rule file is based on Ruta Framework and it helps the system to annotate the text based on the dictionary lookup. For example, it annotates all the text that is found in the anger dictionary as Anger Terms. The position of this term is also captured. All the inputs are fed into the Scoring model to detect the sentence level emotions and also the document level emotion. The document level emotion is returned as the overall emotion at the document level.

The following code is an example of a rule definition.

```
PACKAGE com.ibm.sifs.analytics.emotion.types;

# Sample Rule
# load dictionary
WORDLIST anger_dict = 'anger_dict.txt';
WORDLIST joy_dict = 'joy_dict.txt';

# Declare type definitions
DECLARE Anger;
```

```

DECLARE Joy;

# Detect sentence
DECLARE Sentence;
PERIOD #{ -> MARK(Sentence)} PERIOD;

MARKFAST(Anger, anger_dict, true);
MARKFAST(Joy, joy_dict, true);
# Same for other emotions

```

The emotion detection dictionary

Emotion detection is a java based library and is available as JAR. Currently, it is used in the Real-time Analytics component to detect the emotions in real time and score the emotions in the incoming documents.

As shown in the following diagram, it offers two functions:

- Initialize, which initializes the Emotion library by loading the dictionary and the rules. This function needs to be called only once, and must be started when dictionaries or rules are changed.
- Detect Emotion, which takes text as input and returns a JSON string as a response.

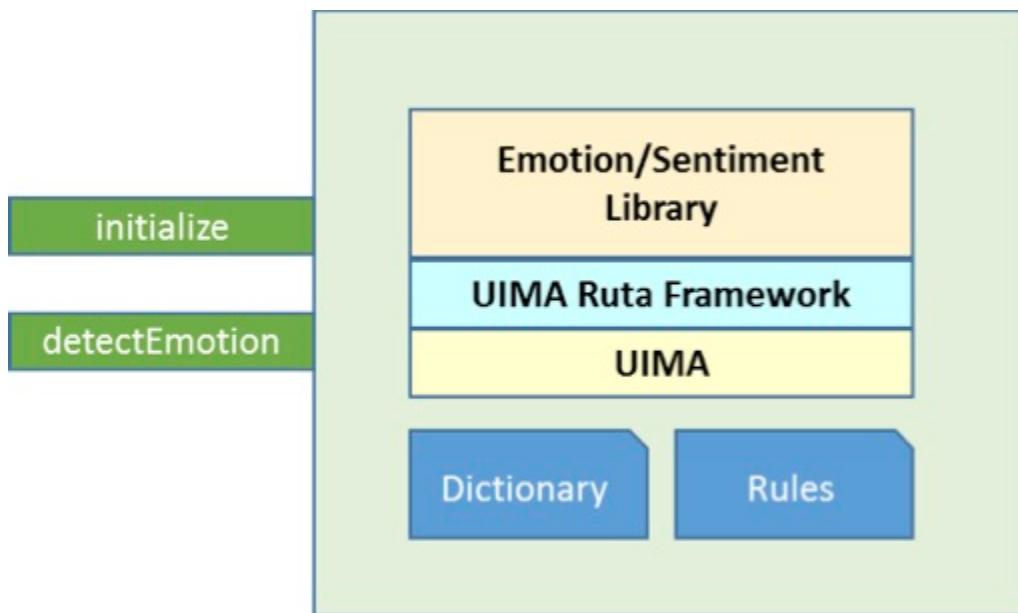


Figure 43: Emotion detection library

Definitions

```
public static void initialize(String dictionaryPath, String rulePath) throws Exception
```

```
public static String detectEmotion(String text)
```

Sample input

```

The investment you made with ABC company stocks are doing pretty good. It has
increased
50 times. I wanted to check with you to see if we can revisit your investment
portfolios for better
investments and make more profit. Please do check the following options and let me
know. I can
visit you at your office or home or at your preferred place and we can discuss on new
business.

```

Market is doing pretty good. If you can make right investments now, it can give good returns on your retirement.

Sample response

```
{
  "sentiment": {
    "score": 1,
    "type": "positive"
  },
  "emotions": {
    "joy": "1.0",
    "sad": "0.0",
    "disgust": "0.0",
    "anger": "0.0",
    "fear": "0.0"
  },
  "keywords": {
    "negative": [],
    "joy": ["pretty", "retirement", "good", "profit"],
    "sad": ["retirement"],
    "disgust": [],
    "anger": [],
    "fear": ["retirement"]
  },
  "status": {
    "code": "200",
    "message": "success"
  }
}
```

Starting the module

```
// Initialize Module (ONLY ONCE)
EmotionAnalyzer.initialize(<path to dictionaries>, <path to rule file>);

// Invoke the library (For every incoming document)
String resultJSON = EmotionAnalyzer.detectEmotion(text);
```

Note: Errors or exceptions are returned in the JSON response under the Status element with a code of 500 and an appropriate message, as shown in the following example.

```
"status": {
  "code": "200",
  "message": "success"
}
```

Concept Mapper library

The Concept Mapper library uses Apache UIMA Ruta (RULE based Text Annotation) to detect the concepts in unstructured text such as emails, instant messages, or voice transcripts.

The library detects the following concepts from the text

- Tickers – Stock Symbol
- Recruit Victims – Evidence of a trader who is trying to get clients to invest in a specific ticker. This activity is indicated as “Recruit Victims.”
- Recruit Conspirators – Evidence of a trader who is collaborating with other traders to conduct a market abuse activity such as “pump/dump”. This activity is indicated as “Recruit Conspirators” in the surveillance context.

Note: If there is more than one ticker in the text, all the tickers are extracted and returned as a comma-separated string.

How the library works

The solution uses a dictionary of tickers, keywords or phrases that represent Recruit Victims and Recruit Conspirators, and concepts and rules to detect the concepts in the text. The dictionaries include the `recruit_conspirators.txt`, `recruit_victims_dict.txt`, and `tickers_dict.txt` files. Each dictionary is a collection of words that represent different concepts in the text.

The rule file is based on Ruta Framework and it helps the system to annotate the text based on the dictionary lookup. For example, it annotates all the text that is found in the Recruit Victims dictionary as Recruit Victims Terms. The position of this term is also captured.

The following code is an example of a rule.

```
PACKAGE com.ibm.sifs.analytics.conceptmapper.types;

# Sample Rule
# Load Dictionary
WORDLIST tickers_dict = 'tickers_dict.txt';
WORDLIST recruit_victims_dict = 'recruit_victims_dict.txt';
WORDLIST recruit_conspirators_dict = 'recruit_conspirators.txt';
WORDLIST negative_dict = 'negative_dict.txt';

# Type definitions
DECLARE Ticker;
DECLARE RecruitConspirators;
DECLARE RecruitVictims;
DECLARE Negative;

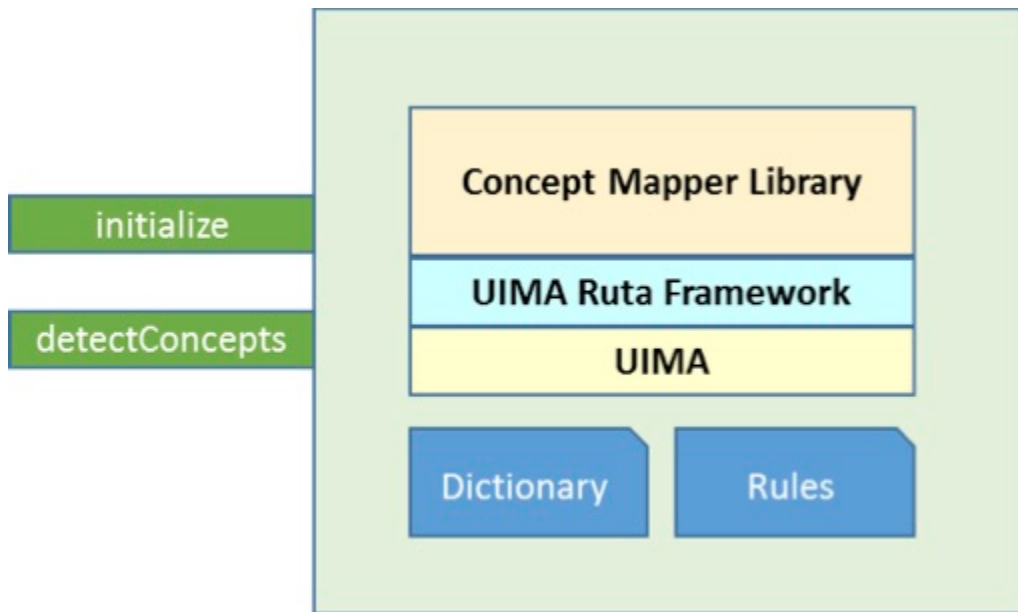
# Annotate/Identify the concepts
MARKFAST(Negative, negative_dict, true);
MARKFAST(Ticker, tickers_dict, false);
MARKFAST(RecruitConspirators, recruit_conspirators_dict, true);
MARKFAST(RecruitVictims, recruit_victims_dict, true);
```

The Concept Mapper dictionary

Concept Mapper is a java-based library and is available as JAR. Currently, it is used in the Real-time analytics component to detect the concepts in real time from the incoming text. As shown in the following diagram, it offers the following functions:

- Initialize, which initializes the library by loading the dictionary and the rules. This function needs to be called only once, and must be started when dictionaries or rules are changed.
- Detect Concepts, which takes text as input and returns a JSON string as a response.

Figure 44: Concept mapper library



Definitions

```
public static void initialize(String dictionaryPath, String rulePath) throws Exception
```

```
public static String detectConcepts(String text)
```

Sample input

I wanted to inform you about an opportunity brought to us by an insider, Mr. Anderson, from ABC Corporation. They specialize in manufacturing drill bits for deep-sea oil rigs. Mr. Anderson owns about 35% of the float and would like us to help increase the price of his company's stock price. If we can help increase the price of the stock by 150%, we would be eligible for a substantial fee and also 1.5% of the profit Mr. Anderson will make disposing the shares at the elevated price. Would you be interested in joining our group in helping Mr. Anderson?

Sample response

```
{
  "concepts": {
    "recruitconspirators": true,
    "tickers": ["ABC"],
    "recruitvictims": false
  },
  "status": {
    "code": "200",
    "message": "success"
  }
}
```

Starting the module

```
// Initialize Module (ONLY ONCE)
ConceptMapper.initialize(<path to dictionaries>, <path to rule file>);
```

```
// Invoke the library (For every incoming document)
String resultJSON = ConceptMapper.detectConcepts(text);
```

Note: Errors or exceptions are returned in the JSON response under the Status element with a code of 500 and an appropriate message, as shown in the following example..

```
"status": {
  "code": "200",
  "message": "success"
}
```

Classifier library

The Classifier library uses MALLET to classify documents into predefined classes and associate probability scores to the classes.

The library can be used to define the following classifications:

- Confidential / Non-confidential documents
- Business / Personal
- News / Announcement / Promotional
- Trading / Non-Trading

How the library works

The Classifier library uses a client/server model. The server library is used to train the model and for the export of the classifier models. The client library uses the classifier model and to classify the incoming documents in real time.

The Classifier library

Classifier is a java based library and is available as JAR. Currently, it is used in the Real-time analytics component to detect the concepts in real time from the incoming text. As shown in the following diagram, it offers the following functions:

- Initialize, which initializes the library by loading the prebuilt classification models. The library can be initialized with multiple classifiers. This function needs to be called only once, and must be started when dictionaries or rules are changed. .
- Classify Docs, which takes text as input and returns a JSON string as a response.

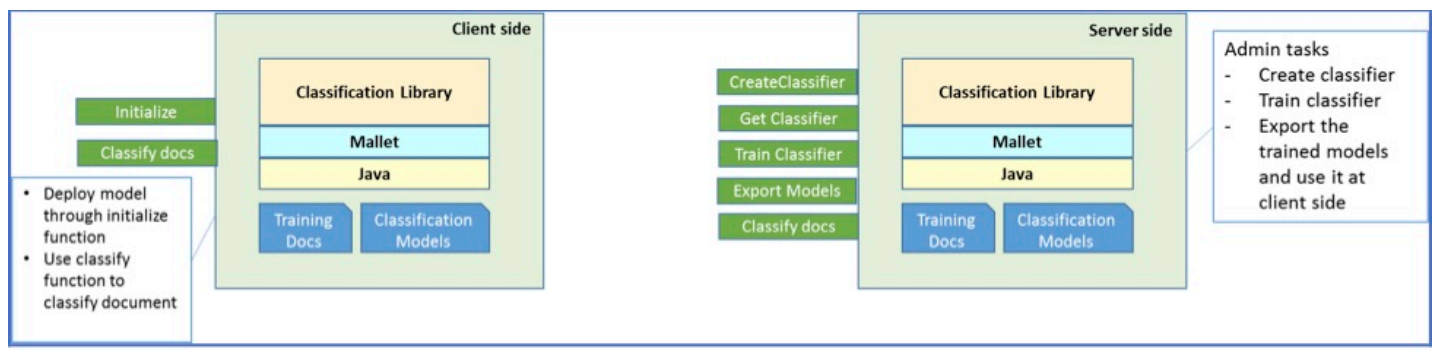


Figure 45: Classifier library

Definitions

```
public static int initialize(Map<String, String> modelMap)
```

```
public static String classify(String classifierName, String document){
```

Sample input

I wanted to inform you about an opportunity brought to us by an insider, Mr. Anderson, from ABC Corporation. They specialize in manufacturing drill bits for deep-sea oil rigs. Mr. Anderson owns about 35% of the float and would like us to help increase the price of his company's stock price. If we can help increase the price of the stock by 150%, we would be eligible for a substantial fee and also 1.5% of the profit Mr. Anderson will make disposing the shares at the elevated price. Would you be interested in joining our group in helping Mr. Anderson?

Sample response

```
{
  "classes": [{
    "confidence": 0.22,
    "class_name": "Confidential"
  }, {
    "confidence": 0.77,
    "class_name": "Non-Confidential"
  }],
  "top_class": "Non-Confidential",
  "status": {
    "code": "200",
    "message": "success"
  }
}
```

Starting the module

```
// Initialize Module (ONLY ONCE)
Map<String, String> modelMap = new HashMap<String, String>();

// testclassifier.cl is the export of the trained model using server library
// "confidentiality" is the name of the initialized classifier
modelMap.put("confidentiality", "/models/testclassifier.cl");

int returnCode = SurveillanceClassifier.initialize(modelMap);

// Invoke the library (For every incoming document)
// "confidentiality" is the name of the initialized classifier
String response = SurveillanceClassifier.classify("confidentiality", text);
```

Note: Errors or exceptions are returned in the JSON response under the Status element with a code of 500 and an appropriate message, as shown in the following example.

```
"status": {
  "code": "200",
  "message": "success"
}
```

Server-side library

The server-side library is RESTful and exposes APIs to operate with the Classifier model. It offers the following services

Table 11: Server-side library

Method	URL	Input	Output
POST	/text/v1/classifiers	JASON Payload	JSON response
GET	/text/v1/classifiers		JSON response
GET	/text/v1/classifiers/{classifierid}		JSON response
DELETE	/text/v1/classifiers/{classifierid}		JSON response
POST	/text/v1/classifiers/{classifierid}/export	Query param: Export model path	JSON response Exported Model

Service details

1. Create classifier

Table 12: Create classifier

Method	URL	Input	Output
POST	/text/v1/classifiers	JASON Payload	JSON response

The service allows users to create and train any number of classifiers. It also allows users to export the trained model to use it from the client side, for example, by using a CURL command to try the POST:

```
curl -k -H 'Content-Type:application/json' -X POST --data-binary @payload.json
http://localhost:9080/analytics/text/v1/classifiers
```

The payload provides details, such as the Classifier name and training data folders for each class in the classifier. The documents need to be available in the server. Currently, the library does not support uploading of training documents.

Note: If the existing classifier name is provided, the classifier overrides it.

The following code is an example JSON payload:

```
{
  "name": "confidential",
  "training-data": [
    { "class": "confidential",
      "trainingdatafolder": "/home/sifs/training_data/confidential" },
    { "class": "non-confidential",
      "trainingdatafolder": "/home/sifs/training_data/non-confidential" }
  ]
}
```

The following code is an example response:

```
{
  "status": {
    "message": "Successfully created Classifier - confidential",
    "status": "200"
  }
}
```

2. Get all classifiers

The service lists the available classifiers in the system.

Table 13: Get all classifiers

Method	URL	Input	Output
POST	/text/v1/classifiers		JSON response

The following code is an example CURL command:

```
curl -k -H 'Content-Type:application/json'  
http://localhost:9080/analytics/text/v1/classifiers
```

The following code is an example response:

```
{  
  "classifiers": ["confidential"],  
  "status": {  
    "message": "Success",  
    "code": "200"  
  }  
}
```

3. Get details on a classifier

The service retrieves the details of the requested classifier.

Table 14: Get details on a classifier

Method	URL	Input	Output
GET	/text/v1/classifiers/ {classifierid}		JSON response

The following code is an example CURL command:

```
curl -k -H 'Content-Type:application/json'  
http://localhost:9080/analytics/text/v1/classifiers/confidential
```

The following code is an example response:

```
{  
  "classifiers": ["confidential"],  
  "status": {  
    "message": "Success",  
    "code": "200"  
  }  
}
```

4. Delete Classifier

This service deletes the requested classifier from the library.

Table 15: Delete classifier

Method	URL	Input	Output
DELETE	/text/v1/classifiers/ {classifierid}		JSON response

The following code is an example CURL command:

```
curl -k -X DELETE http://localhost:9080/analytics/text/v1/classifiers/confidential/
```

The following code is an example response:

```
{ "status": { "message": "Classifier - confidential is successfully deleted", "code": "200" } }
```

5. Export Classification Model

The service exports the model file for the classifier. The model file can be used on the client side. It is a serialized object and it can be deserialized on the client side to create the classifier instance and classify the documents.

Table 16: Export classification model			
Method	URL	Input	Output
POST	/text/v1/classifiers/{classifierid}/export	Query param: Export model path	JSON response, Exported Model

The following code is an example CURL command:

```
curl -k -X POST
http://loext/v1/classifiers/confidential/export/?exportmodelpath=/home/sifs/classifiers
```

The following code is an example response:

```
{
  "status": {
    "message": "Classification Model successfully exported /home/sifs/classifiers",
    "code": "200"
  }
}
```

Chapter 7. Inference engine

IBM Surveillance Insight for Financial Services contains an implementation of a Bayesian inference engine that helps in deciding whether an alert needs to be created for a set of evidences available on a specific date.

The inference engine takes the risk model (in JSON format) and the scores for each risk indicator that is referred to by the risk model as input. The output of the engine is a JSON response that gives the scores of the risk indicators and the conclusion of whether an alert needs to be created.

Inference engine risk model

The IBM Surveillance Insight for Financial Services inference engine risk model is represented in JSON format.

The following code is a sample.

```
{
  "edges": [{
    "source": "107",
    "target": "15"
  }, {
    "source": "108",
    "target": "15"
  }, {
    "source": "105",
    "target": "108"
  }, {
    "source": "106",
    "target": "108"
  }, {
    "source": "109",
    "target": "107"
  }
],
  "nodes": [{
    "parents": [],
    "desc": "Party Past history",
    "lookupcode": "TXN05",
    "outcome": ["low", "medium", "high"],
    "threshold": [0.33, 0.75],
    "id": "105",
    "category": "Party",
    "level": 2,
    "source": "TRADE",
    "subcategory": "NO_DISPLAY",
    "name": "Party Past History",
    "probabilities": [0.33, 0.33, 0.33],
    "x": 13.5,
    "y": 159
  }, {
    "parents": [],
    "desc": "Counterparty Past history",
    "lookupcode": "TXN06",
    "outcome": ["low", "medium", "high"],
    "threshold": [0.33, 0.75],
    "id": "106",
    "category": "Party",
    "level": 2,
    "source": "TRADE",
    "subcategory": "NO_DISPLAY",
    "name": "Counterparty Past History",
    "probabilities": [0.33, 0.33, 0.33],
    "x": 188.5,
```

```

    "y": 132
  }, {
    "parents": [],
    "desc": "Transaction Deal Rate Anomaly",
    "lookupcode": "TXN09",
    "outcome": ["low", "medium", "high"],
    "threshold": [0.33, 0.75],
    "id": "109",
    "category": "Transaction",
    "level": 2,
    "source": "TRADE",
    "subcategory": "NO_DISPLAY",
    "name": "Transaction Deal Rate Anomaly",
    "probabilities": [0.33, 0.33, 0.33],
    "x": 440.5,
    "y": 117
  }, {
    "parents": ["109"],
    "desc": "Transaction Risk",
    "lookupcode": "TXN07",
    "outcome": ["low", "medium", "high"],
    "rules": ["RD109 == 'low'", "RD109 == 'medium'", "RD109 == 'high'"],
    "threshold": [0.33, 0.75],
    "id": "107",
    "category": "Derived",
    "level": 1,
    "source": "TRADE",
    "subcategory": "NO_DISPLAY",
    "name": "Transaction Risk",
    "probabilities": [1, 0, 0, 0, 1, 0, 0, 0, 1],
    "x": 443,
    "y": 242
  }, {
    "parents": ["105", "106"],
    "desc": "Involved Party Risk",
    "lookupcode": "TXN08",
    "outcome": ["low", "medium", "high"],
    "rules": ["RD105 == 'low' && RD106 == 'low'", "(RD105 == 'medium' && RD106 == 'low') || (RD105 == 'low' && RD106 == 'medium') || (RD105 == 'medium' && RD106 == 'medium') || (RD105 == 'high' && RD106 == 'low') || (RD105 == 'low' && RD106 == 'high')", "(RD105 == 'high' && RD106 == 'high') || (RD105 == 'high' && RD106 == 'medium') || (RD105 == 'medium' && RD106 == 'high')"],
    "threshold": [0.33, 0.75],
    "id": "108",
    "category": "Derived",
    "level": 1,
    "source": "TRADE",
    "subcategory": "NO_DISPLAY",
    "name": "Involved Party Risk",
    "probabilities": [1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1],
    "x": 126,
    "y": 247
  }, {
    "parents": ["107", "108"],
    "desc": "Risk",
    "lookupcode": "RD15",
    "outcome": ["low", "medium", "high"],
    "rules": ["RD107 == 'low' && RD108 == 'low'", "(RD107 == 'medium' && RD108 == 'low') || (RD107 == 'low' && RD108 == 'medium') || (RD107 == 'medium' && RD108 == 'medium') || (RD107 == 'high' && RD108 == 'low') || (RD107 == 'low' && RD108 == 'high')", "(RD107 == 'high' && RD108 == 'high') || (RD107 == 'high' && RD108 == 'medium') || (RD107 == 'medium' && RD108 == 'high')"],
    "threshold": [0.33, 0.75],
    "id": "15",
    "category": "Derived",

```

```

        "level": 0,
        "source": "COMM",
        "subcategory": "NO_DISPLAY",
        "name": "Risk",
        "probabilities": [1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
0, 0, 0, 1, 0, 0, 1],
        "x": 290,
        "y": 424
    }
}

```

The content of the risk model is a network of nodes and edges. The nodes represent the different risk indicators and the edges represent the link between the risk indicators nodes and the aggregated risk indicator nodes.

The risk indicator look-up codes that are mentioned in the risk_indicator_master table are used to refer to specific risk indicators.

Each node also has an X,Y co-ordinate for the placement of the node in the user interface.

If you create a new model for a use case, the model must be updated in the risk_model_master table. The risk_model_master table contains one row per use case.

Run the inference engine

You can run the inference engine as a Java API.

```
String sifs.analytics.inference.BayesianInference.runInference(String network, String
evidenceJSON)
```

network refers to the risk model JSON that must be used for the use case.

evidenceJSON refers to the JSON that contains the risk indicators and their scores that are applied on the risk model.

The following code is an example of the JSON.

```

{
  "nodes": [{
    "id": "1",
    "value": 0.5
  },
  {
    "id": "2",
    "value": 0.6
  }
]
}

```

The result is a string in JSON format that contains the outcome of the inference engine. The following is an example of a response.

```

{
  "result": [{
    "score": 1,
    "id": "33"
  }, {
    "score": 0.9,
    "id": "34"
  }, {
    "score": 0.7135125,
    "id": "35"
  }, {
    "score": 0.15500289242473736,
    "id": "36"
  }, {

```

```

        "score": 0,
        "id": "37"
    }, {
        "score": 0.75,
        "id": "15"
    }, {
        "score": 1,
        "id": "26"
    }, {
        "score": 0.33,
        "id": "27"
    }, {
        "score": 0.9,
        "id": "30"
    }, {
        "score": 0,
        "id": "31"
    }, {
        "score": 0.887188235294118,
        "id": "32"
    }
  ],
  "alert": {
    "isAlert": true,
    "score": 0.75
  },
  "status": {
    "code": 200,
    "message": "success"
  }
}

```

The `isAlert` field gives the outcome of whether an alert needs to be created.

The `score` field contains the alert score that is consolidated from the individual risk indicator scores.

The `status` field indicates the overall status of the inference.

The inference engine returns whether the inputs are risky enough to cause an alert. Whether a new alert needs to be created or an existing alert needs to be updated depends on the use case implementation. The inference engine does not consider new alert or updating an existing alert.

Chapter 8. Indexing and searching

IBM Surveillance Insight for Financial Services provides indexing capabilities for analytics results.

Surveillance Insight uses IBM Solr for indexing and searching features. The search capabilities allow users to search for any kind of communication (email, chat, or voice message), by using keywords or other attributes. Indexing is performed on the content of each communication regardless of the channels and on the results of the analytics.

Indexing is performed on three objects:

- Alert
- E-Comm (includes email, chat, and voice)
- Employee

Employee data is indexed as a one-time initial load. E-Comm content and the analytic results are indexed. The email body, collated chat content, and all utterances of voice speech are indexed.

When an alert is created, certain attributes of the alert are also indexed. Updates to an alert, such as adding new evidence to an existing alert is updated in Solr.

Solr schema for indexing content

S.no	Field Name	Field Type	Description	Multi-Valued	Indexed	Stored
1	activealerts	Int	Number of active alerts on employee	N	N	Y
2	alertevidenceids	String	Evidenceids of an alert	Y	Y	Y
3	alertid	String	Alert ID from SIFS System	Y	Y	Y
4	alertstatus	String	Alert Status	N	Y	Y
5	alerttype	String	Alert type as defined in SIFS	N	Y	Y
6	assetclass	String	Asset class of the ticker	N	Y	Y
7	channel	String	Channel of E-Comm, like email, chat or voice	N	Y	Y
8	city	String	Employee's City	N	Y	Y
9	classification	String	Classification of E-Comm content	Y	Y	Y
10	concepts	String	Concepts identified in E-Comm content	Y	Y	Y
11	content	String	Content of E-Comm	N	Y	N
12	description	String	Description assigned to a E-Comm	N	Y	Y

13	doctype	String	Type of document. Has values of alert, ecomm or party.	N	Y	Y
14	ecommid	Int	E-Comm ID from SIFS System	N	Y	Y
15	entity	String	Entities as identified by Entity extractor on E-Comm content	Y	Y	Y
16	Employeeid	String	Employee ID from SIFS System	N	Y	Y
17	evidenceids	String	Evidence IDs as identified by feature extractors on E-Comm content	Y	N	Y
18	id	String	ID of the object	N	Y	Y
19	Name	String	E-Comm initiators name	N	Y	Y
20	participants	String	Partyids of Participants of an E-Comm	Y	Y	Y
21	partyid	String	Party ID of initiator of an E-Comm	N	Y	Y
22	pastviolations	Int	Number of past violations	N	N	Y
23	Riskrating	Float	Risk rating of an employee	N	N	Y
24	role	String	Role an employee	Y	Y	Y
25	state	String	Employee's State	N	Y	Y
26	tags	String	Tags assigned to an E-Comm	Y	Y	Y
27	ticker	String	Ticker identified relevant to E-Comm or Alert.	N	Y	Y
28	Tradernames	String	Participants Names of an E-Comm	Y	Y	Y

Chapter 9. API reference

This section provides information for the APIs that are provided with Surveillance Insight for Financial Services.

Alert service APIs

The following API services are available for the Alert service from the SIFSServices application that is deployed to WebSphere Application Server.

Update Communication Tags

REST method

```
PUT
```

REST URL

```
/alertservice/v1/alert/{alertid}/commevidences/{evidenceid}
```

Request body

```
{
  "tags": [
    "anger",
    "sad",
    "disgust",
    "fear",
    "happy",
    "negative",
    "positive"
  ]
}
```

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Update Alert Action

REST method

```
PUT
```

REST URL

```
/alertservice/v1/alert/action/{alertid}
```

Request body

```
{
  "alertactionstatus": "agree",
  "alertactioncomments": "reason for agreeing"
}
```

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Get Top Alerts

REST method

GET

REST URL

/alertservice/v1/alert/alerts

This service returns alerts that are not closed and note rejected. It returns alerts for the maximum number of dates available in the database. For example, 30 days.

Response body

```
{
  "alerts": [
    {
      "date": "04/04/2017",
      "ticker": "TCV",
      "alertscore": 0.789,
      "enddate": null,
      "assigned": "",
      "alertid": 10000081,
      "type": "spoofing",
      "startdate": null,
      "assetclass": "Equity",
      "status": "CLOSED"
    },
    {
      "date": "04/04/2017",
      "ticker": "KVQ /FSB /EYS /HAO",
      "alertscore": 0.521,
      "enddate": null,
      "assigned": "",
      "alertid": 1000005,
      "type": "spoofing",
      "startdate": null,
      "assetclass": "Derivatives/Derivatives/Interest rates/Interest rates",
      "status": "CLOSED"
    }
  ]
}
```

Get Alert

REST method

GET

REST URL

/alertservice/v1/alert/{alertid}?forHeader=<true/false>

If forHeader=true, then limited information (main alert details and involved parties details) related to alert are provided in the output JSON. For no value or any other value of forHeader, the complete JSON is sent as the output.

Response body

Get Communication Evidence

REST method

GET

REST URL

/alertservice/v1/alert/{alertid}/commevidences/{evidenceid}

Response body

```
{
  "date": "07/17/2015",
  "partyname": "Jaxson Armstrong",
  "evidenceid": 3022,
  "communicationDetails": {
    "date": "07/17/2015",
    "communicationId": 653,
    "initiator": "Jaxson_Armstrong@digitbrokerage.com",
    "communicationType": null,
    "time": "04:33:13",
    "participants":
    "John_Collins@digitbrokerage.com,Isabella_Brown@digitbrokerage.com"
  },
  "riskscore": 0.7071067811865476,
  "description": "Mass communications sent by Jaxson Armstrong to influence investors during the for today",
  "riskindicatorid": 0,
  "type": "e-mail",
  "partyid": 10002,
  "mentioned": "Corporation, FDA, PDZ, PDZ Corporation"
}
```

Get Network Details

REST method

GET

REST URL

/alertservice/v1/alert/{alertid}/network

Response body

```
{
  "network": [
    {
      "entitytype": "mentions",
      "source": {
        "entitytype": "people",
        "entityname": "Kimberly Johnst",
        "partyimage": "/some/location/of/IHS/server/0.png",
        "entityid": 100120
      },
      "evidences": [
        {

```

```

        "date": "04/30/2017",
        "evidenceid": 1000004,
        "riskscore": 0.6301853860302291,
        "description": "Savannah Brooks communicated in Unusual timings  for
today.",
        "type": "e-mail"
    }
],
"target": {
    "entitytype": "people",
    "entityname": "Gabriel Phillips",
    "partyimage": "/some/location/of/IHS/server/0.png",
    "entityid": 100100
}
},
{
    "entitytype": "mentions",
    "source": {
        "entitytype": "people",
        "entityname": "Kimberly Johnst",
        "partyimage": "/some/location/of/IHS/server/0.png",
        "entityid": 100120
    },
    "evidences": [
        {
            "date": "04/30/2017",
            "evidenceid": 1000004,
            "riskscore": 0.6301853860302291,
            "description": "Savannah Brooks communicated in Unusual timings  for
today.",
            "type": "e-mail"
        }
    ],
    "target": {
        "entitytype": "people",
        "entityname": "Natalie Cook",
        "partyimage": "/some/location/of/IHS/server/0.png",
        "entityid": 100134
    }
}
]
}

```

Get Print Preview (PDF)

REST method

GET

REST URL

/alertservice/v1/alert/{alertid}/print

Get All Feature Tags

REST method

GET

REST URL

/alertservice/v1/alert/alltags

Response body

```
[
  {
    "category": "Emotion",
    "tags": [
      "anger",
      "sad",
      "disgust",
      "fear",
      "happy",
      "negative",
      "positive"
    ]
  },
  {
    "category": "Concepts",
    "tags": [
      "Recruit Victims",
      "Recruit Conspirators"
    ]
  },
  {
    "category": "Classification",
    "tags": [
      "Confidential",
      "Non-Confidential",
      "Personal",
      "Business",
      "Promotional",
      "News",
      "Announcement",
      "Trading",
      "Non-Trading"
    ]
  }
]
```

Create Alert

REST method

POST

REST URL

/alertservice/v1/alert/createAlert

In the input JSON:

- The Party role can be any text.
- The source must be SIFS unless another valid alert source exists in the SIFS ALERT_SOURCE_MASTER table.
- Risk indicators are optional. If they are provided, the look-up codes should be valid from the RISK_INDICATOR_MASTER table.
- The Alert id must be generated uniquely by the application that is generating the alert.
- COMMID is optional.
- All ids (except for the Alert id) should be valid in the Surveillance Insight database during the alert creation. Otherwise, the alert creation fails.

Request body

```
{
  "alert": {
```

```

"timestamp": "04/22/2017 14:12:12",
"involvedparties": [
  {
    "role": "trader",
    "partyid": 10008
  },
  {
    "role": "counter party",
    "partyid": 10009
  }
],
>alertscore": 0.63,
"ticker": "IQN",
"startdate": "04/22/2017",
"source": "SIFS",
"status": "NEW",
>alertid": 209,
"riskindicators": [
  {
    "riskscore": 0.07,
    "riskindicator": "RD6"
  },
  {
    "riskscore": 0.008,
    "riskindicator": "RD5"
  }
],
"enddate": "04/22/2017",
"type": "Spoofing",
"evidences": [
  {
    "evidenceid": 14961
  },
  {
    "commid": 10000,
    "evidenceid": 14962
  }
]
}

```

Response body

```
{ "message": "Alert Created", "status": 200 }
```

OR

```
{ "message": "Alert Creation Failed. Please check the server logs for more details.", "status": 500 }
```

 for failure

Update Alert

REST method

POST

REST URL

/alertservice/v1/alert/updateAlert

The profit field indicates the profit that is made by the primary party that is involved in the alert. It is an optional field.

The UPDATE_RI_SCORES flag indicates that the risk indicator scores must be updated in the alert_risk_indicator_score table in the Surveillance Insight database. The default value is false. This means that if this field is not provided, the service creates new risk indicators for the specified dates.

Risk indicators are optional. If they are provided, the look-up codes should be valid from the RISK_INDICATOR_MASTER table.

All ids should be valid in the Surveillance Insight database during the alert creation. Otherwise, the alert creation fails.

Evidence id is optional. If evidence id is not provided and the rest of the evidence details are provided, the service creates the evidence and associate it with the alert. If the evidence id is provided, the rest of the fields in the evidence tag are ignored. The evidence id is used to update the alert_evidence_rel table.

Commid is optional.

Request body

```
{
  "alert": {
    "timestamp": "04/10/2017 14:23:12",
    "profit": 729.01,
    "alertscore": 0.45,
    "UPDATE_RI_SCORES": "true",
    "alertid": 208,
    "riskindicators": [
      {
        "riskscore": 0.999,
        "riskindicator": "RD14"
      },
      {
        "riskscore": 0.999,
        "riskindicator": "RD15"
      }
    ],
    "enddate": "05/19/17",
    "evidences": [
      {
        "timestamp": "09/12/17 16:12:03",
        "involvedparties": [
          {
            "partyid": "10002"
          },
          {
            "traderid": "10003"
          }
        ],
        "tickers": "IQN,MSL",
        "commid": 10000,
        "evidenceid": 320,
        "description": "some description",
        "riskscore": 0.9,
        "riskindicator": "Risk",
        "type": "trade"
      },
      {
        "evidenceid": 319
      }
    ],
    "partyid": "10001"
  }
}
```

Response body

```
{"message":"Alert <alert id> has been updated successfully ","status":200}
```

OR

```
{"message":"Alert Updation Failed for <alert id>","status":500} for failure
```

Create Evidence

REST URL

```
/alertservice/v1/alert/creteEvidence
```

Request body

```
{
  "alert": {
    "evidences": [
      {
        "timestamp": "10/22/2017 16:12:03",
        "involvedparties": [
          {
            "role": "trader",
            "partyid": 10030
          },
          {
            "partyid": 10026
          }
        ],
        "tickers": "IQN",
        "description": "some description",
        "riskscore": 0.22,
        "riskindicator": "RD2",
        "type": "Trade"
      },
      {
        "timestamp": "09/23/2017 16:12:03",
        "involvedparties": [
          {
            "partyid": 10032
          }
        ],
        "description": "some description",
        "riskscore": 0.3,
        "riskindicator": "Risk",
        "type": "Trade"
      }
    ]
  }
}
```

Response body

```
{"message":"{\"evidences\": [{\"evidenceid\":21624},{\"evidenceid\":21625}]}","status":200}
```

(ids provided above are samples, actual id may vary)

OR

```
{"message":"Evidence Creation Failed","status":500}
```

Notes service APIs

The following API services are available for the Notes service from the SIFSServices application that is deployed to WebSphere Application Server.

Create note

REST method

```
POST
```

REST URL

```
/noteservice/v1/alert/{alertid}/note
```

Request body

```
{
  "note": {
    "notetitle" : "some title",
    "content": "3 Note added for testing",
    "attachment": "<binary data>",
    "authorid": 10000,
    "reference": "Overview Page"
  }
}
```

Response body

```
{
  "noteid": 77,
  "message": "Success",
  "status": 200
}
```

Update note

REST method

```
PUT
```

REST URL

```
/noteservice/v1/alert/{alertid}/note/{noteid}
```

Request body

```
{
  "note": {
    "notetitle" : "some title",
    "content": "update note for note 3",
    "authorid": 10000,
    "reference": "Overview Page"
  }
}
```

Response body

```
{
  "message": "Success",
}
```

```
{
  "status": 200
}
```

Delete note

REST method

DELETE

REST URL

/noteservice/v1/alert/{alertid}/note/{noteid}

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Add attachment

REST method

POST

REST URL

/noteservice/v1/alert/{alertid}/note/{noteid}/attachment

Request body

```
{
  "note": {
    "attachment": "<binary data>"
  }
}
```

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Delete attachment

REST method

DELETE

REST URL

/noteservice/v1/alert/{alertid}/note/{noteid}/attachment

Response body

```
{
  "message": "Success",
}
```

```
"status": 200
}
```

Get all notes

REST method

```
GET
```

REST URL

```
/noteservice/v1/alert/{alertid}/notes
```

Response body

```
{
  "notes": [
    {
      "date": "02/17/2017",
      "author": "Chris Brown",
      "noteid": 1,
      "time": "16:23:59",
      "content": "Some note",
      "attachment": "<binary data>",
      "reference": "Overview Page"
    },
    {
      "date": "02/17/2017",
      "author": "Chris Brown",
      "noteid": 3,
      "time": "17:02:39",
      "content": "More note",
      "attachment": "<binary data>",
      "reference": "Overview Page"
    }
  ]
}
```

Get note

REST method

```
GET
```

REST URL

```
/noteservice/v1/alert/{alertid}/note/{noteid}
```

Response body

```
{
  "note": {
    "date": "02/17/2017",
    "author": "Chris Brown",
    "noteid": 1,
    "time": "16:23:59",
    "content": "some note",
    "attachment": "<binary data>",
    "reference": "Overview Page"
  }
}
```

Get party ID

REST method

GET

REST URL

/surveillanceui/v1/party/{employeeid}

Response body

```
{
  "partyid": 12345
}
```

Get top parties

REST method

GET

REST URL

/surveillanceui/v1/party/topParties

Response body

```
{
  "parties": [
    {
      "pastviolations": 0,
      "phone": "(+1)-388-514-2105",
      "partyname": "Annabelle Cole",
      "email": "Annabelle_Cole@digitbrokerage.com",
      "location": "New York, NY",
      "riskrating": 0.0,
      "activealerts": 0,
      "role": "Trader",
      "partyimage": "",
      "partyid": 10047
    }
  ]
}
```

Get party profile

REST method

GET

REST URL

/surveillanceui/v1/party/{partyid}/profile

Response body

```
{
  "partyprofile": {
    "pastviolations": 0,
    "supervisor": {
      "phone": "(+1)-563-745-2070",
      "email": "Aiden_Robinson@digitbrokerage.com",
      "name": "Aiden Robinson"
    },
    "riskrating": 0.0,
    "activealerts": 0,
    "personalinfo": {
      "postalcode": "500090",
      "phone": "(+1)-629-155-1326",

```

```
"employeeid":"10034",
"email":"Jeremiah_Bryant@digitbrokerage.com","street":"PASEO LOS POCITOS",
"deskno":10096,
"state":"NY",
"role":"Trader",
"city":"New York"},
"lastname":"Bryant",
"firstname":"Jeremiah",
"partyimage":"","
"partyid":10034}}
```

Get party alert summary

REST method

GET

REST URL

/surveillanceui/v1/party/{partyid}/summary

Response body

```
{
  "party": {
    "partyname": "Jeremiah Bryant",
    "alertsummary": [
      {
        "status": "closed",
        "count": 30
      },
      {
        "status": "open",
        "count": 40
      },
      {
        "status": "in progress",
        "count": 70
      },
      {
        "status": "manually escalated",
        "count": 80
      },
      {
        "status": "rejected",
        "count": 90
      },
      {
        "status": "automatically escalated",
        "count": 70
      }
    ],
    "alerthistory": [
      {
        "alertId": "1234",
        "score": 0.85,
        "date": "11/24/15",
        "type": "Pump and Dump",
        "status": "New"
      }
    ],
    "partyid": 10034
  }
}
```

Get party anomaly

REST method

GET

REST URL

```
/surveillanceui/v1/party/{partyid}/anomaly
```

```
/surveillanceui/v1/party/{partyid}/anomaly?startDate=yyyy-mm-dd&endDate=yyyy-mm-dd
```

When the start and end dates are not provided, the service returns the data for the last 10 dates in the database for the specific party.

Response body

```
{
  "commanomalies": [{
    "anomalies": [{
      "id": 6,
      "name": "Confidential",
      "score": 0.7071
    }],
    "date": "04/05/2017"
  }]
}
```

Party service APIs

The following API services are available for the Notes service from the SIFSServices application that is deployed to WebSphere Application Server.

Get Party Id

REST method

```
GET
```

REST URL

```
/surveillanceui/v1/party/{employeeid}
```

Response body

```
{
  "partyid": 12345
}
```

Get Top Parties

REST method

```
GET
```

REST URL

```
/surveillanceui/v1/party/topParties
```

Response body

```
{"parties": [
  {
    "pastviolations": 0,
    "phone": "(+1)-388-514-2105",
```

```
"partyname":"Annabelle Cole",
"email":"Annabelle_Cole@digitbrokerage.com",
"location":"New York, NY",
"riskrating":0.0,
"activealerts":0,
"role":"Trader",
"partyimage":"",
"partyid":10047}
]}
```

Get Party Profile

REST method

GET

REST URL

/surveillanceui/v1/party/{partyid}/profile

Response body

```
{"partyprofile":
{"pastviolations":0,
"supervisor":{"phone":"+1-563-745-2070",
"email":"Aiden_Robinson@digitbrokerage.com","name":"Aiden Robinson"},
"riskrating":0.0,
"activealerts":0,
"personalinfo":
{"postalcode":"500090",
"phone":"+1-629-155-1326",
"employeeid":"10034",
"email":"Jeremiah_Bryant@digitbrokerage.com","street":"PASEO LOS POCITOS",
"deskno":10096,
"state":"NY",
"role":"Trader",
"city":"New York"},
"lastname":"Bryant",
"firstname":"Jeremiah",
"partyimage":"",
"partyid":10034}}
```

Get Party Alert Summary

REST method

GET

REST URL

/surveillanceui/v1/party/{partyid}/summary

Response body

```
{"party":
{"partyname":"Jeremiah Bryant",
"alertsummary":[{
"status":"closed",
"count":30
}, {
"status":"open",
```

```

    "count": 40
  }, {
    "status": "in progress",
    "count": 70
  }, {
    "status": "manually escalated",
    "count": 80
  }, {
    "status": "rejected",
    "count": 90
  }, {
    "status": "automatically escalated",
    "count": 70
  }
],
"alerthistory": [{
  "alertId": "1234",
  "score": 0.85,
  "date": "11/24/15",
  "type": "Pump and Dump",
  "status": "New"
}],
"partyid": 10034
}
}

```

Get Party Anomaly

REST method

GET

REST URL

/surveillanceui/v1/party/{partyid}/anomaly

/surveillanceui/v1/party/{partyid}/anomaly?startDate=yyyy-mm-dd&endDate=yyyy-mm-dd

When the start and end dates are not provided, the service returns the data for the last 10 dates in the database.

Response body

```

{
  "commanomalies": [{
    "anomalies": [{
      "id": 6,
      "name": "Confidential",
      "score": 0.7071
    }],
    "date": "04/05/2017"
  }]
}

```

CommServices APIs

The following services are available from `com.ibm.sifs.service.data-2.0.0-SNAPSHOT.war` that is deployed to WebSphere Application Server.

Publish email

REST method

POST

REST URL

/data/email

Request body

A sample email xml is available [here](http://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/samplefile/SurveillanceInsightSampleEcommEmail.xml). (www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/samplefile/SurveillanceInsightSampleEcommEmail.xml)

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Example

```
curl -k -H 'Content-Type: text/plain' -H 'source:Actiance' -X POST --data-binary @email.txt -v --user actiance1:actpwd1 --digest https://localhost:443/CommServices/data/email
```

Example of retrieving an email

```
curl -k -H 'Content-Type: application/json' -H 'source:Actiance' -X GET -v --user actiance1:actpwd1 --digest https://localhost:443/CommServices/data/commevidence/1
```

Publish chat

REST method

POST

REST URL

/data/chat

Request body

A sample chat xml is available [here](http://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/samplefile/SurveillanceInsightSampleEcommChat.xml). (www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/samplefile/SurveillanceInsightSampleEcommChat.xml)

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Example

```
curl -k -H 'Content-Type: text/plain' -H 'source:Actiance' -X POST --data-binary @chat.txt -v --user actiance1:actpwd1 --digest https://localhost:443/CommServices/data/chat
```

Get Comm evidence

REST method

GET

REST URL

```
/data/commevidence/{communicationid}
```

Response body

```
{
  "subject": "XYZ Corporation to develop own mobile payment system",
  "body": "Drugmaker TAM Corporation snapped up rights to a promising cell therapy developed by French biotech firm Collectis to fight blood cancers.\\r\\n\\r\\nThe so-called CAR T cell technology used by Collectis involves reprogramming immune system cells to hunt out cancer. The \"off-the-shelf\" approach recently proved very successful in the case of a baby whom doctors thought almost certain to die.\\r\\n\\r\\nCollectis said on Thursday that TAM Corporation had exercised an option to acquire the exclusive worldwide rights to UCART19, which is about to enter initial Phase I clinical tests, and they would work on the drug's development.\\r\\n\\r\\nCollectis will get $3.2 million upfront from TAM Corporation and is eligible for over $30 million in milestone payments, research financing and royalties on sales. Detailed financial terms for the agreement were not disclosed.\\r\\n\\r\\nUCART19 is being tested for chronic lymphocytic leukaemia and acute lymphoblastic leukaemia. Collectis is developing Chimeric Antigen Receptor T-cell, or CAR-T, immunotherapies using engineered cells from a single donor for use in multiple patients.\""}

```

Policy service APIs

The following services are available from `com.ibm.sifs.service.data-2.0.0-SNAPSHOT.war` that is deployed to WebSphere Application Server.

Create policy

REST method

POST

REST URL

/ecomm/policy

Request body

```
{
  "policy": {
    "policyname": "Policy 2",
    "policycode": "POL2",
    "policytype": "role",
    "policysubcategory": "Sub2",
    "policydescription": "Role Level Policy",
    "role": [
      "Trader",
      "Banker"
    ],
    "features": [{
      "name": "document classifier"
    }, {
      "name": "concept mapper"
    }, {
      "name": "entity extractor"
    }]
  }
}
```

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Example

```
curl -k -H 'Content-Type: application/json' -H 'source:Actiance' -X POST
--data-binary @/home/vrunda/files/policy1_sys.json -v --user actiance1:actpwd1
--digest https://localhost:443/surveillance/ecomm/policy
```

Update policy

REST method

PUT

REST URL

/ecomm/policy

Request body

```
{
  "policy": {
    "policyname": "Policy 1",
    "policycode": "POL1",
    "policytype": "system",
    "policysubcategory": "Sub1",
    "policydescription": "System Policy 1",
    "features": [{
      "name": "emotion"
    }]
  }
}
```

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Example

```
curl -k -H 'Content-Type: application/json' -H 'source:Actiance' -X PUT
--data-binary @/home/vrunda/files/policy1_sys.json -v --user actiance1:actpwd1
--digest https://localhost:443/surveillance/ecommm/policy
```

Activate policy

REST method

PUT

REST URL

/ecommm/policy/{policyCode}

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Example

```
curl -k -H 'Content-Type: application/json' -H 'source:Actiance' -X PUT -v
--user actiance1:actpwd1 --digest https://localhost:443/CommServices/ecommm/policy/
activate/POL1
```

Deactivate policy

REST method

PUT

REST URL

/ecommm/policy/{policyCode}

Response body

```
{
  "message": "Success",
  "status": 200
}
```

Example

```
curl -k -H 'Content-Type: application/json' -H 'source:Actiance' -X PUT -v
--user actiance1:actpwd1 --digest https://localhost:443/CommServices /ecommm/policy/
deactivate/POL1
```

Get policy

REST method

GET

REST URL

/ecomm/policy/{policyCode}

Response body

```
{
  "policy": {
    "policyname": "Policy 1",
    "policycode": "POL1",
    "policytype": "system",
    "policysubcategory": "Sub1",
    "policydescription": "System Policy 1",
    "features": [
      {
        "name": "emotion"
      }
    ]
  }
}
```

Example

```
curl -k -H 'Content-Type: application/json' -H 'source:Actiance' -X GET -v
--user actiance1:actpwd1 --digest https://localhost:443/CommServices/ecomm/policy/
POL1
```

Get all policies

REST method

GET

REST URL

/ecomm/policy

Response body

```
[{"policytype":"system","features":[{"name":"emotion"}],"policysubcategory":
"Sub1","policycode":"POL1","policydescription":"System Policy
1","policyname":"Policy 1"},
{"policytype":"role","features":[{"name":"document classifier"}, {"name":"concept
mapper"},
{"name":"entity extractor"}],"policysubcategory":"Sub2","role":["Trader","Banker"],
"policycode":"POL2","policydescription":"Role Level Policy","policyname":"Policy
2"}]
```

Example

```
curl -k -H 'Content-Type: application/json' -H 'source:Actiance' -X GET -v
--user actiance1:actpwd1 --digest https://localhost:443/CommServices/ecomm/policy
```


Chapter 10. Develop your own use case

The IBM Surveillance Insight for Financial Services platform allows you to develop your own use case implementations. The following diagram shows the series of steps that you must follow to develop your own use cases.

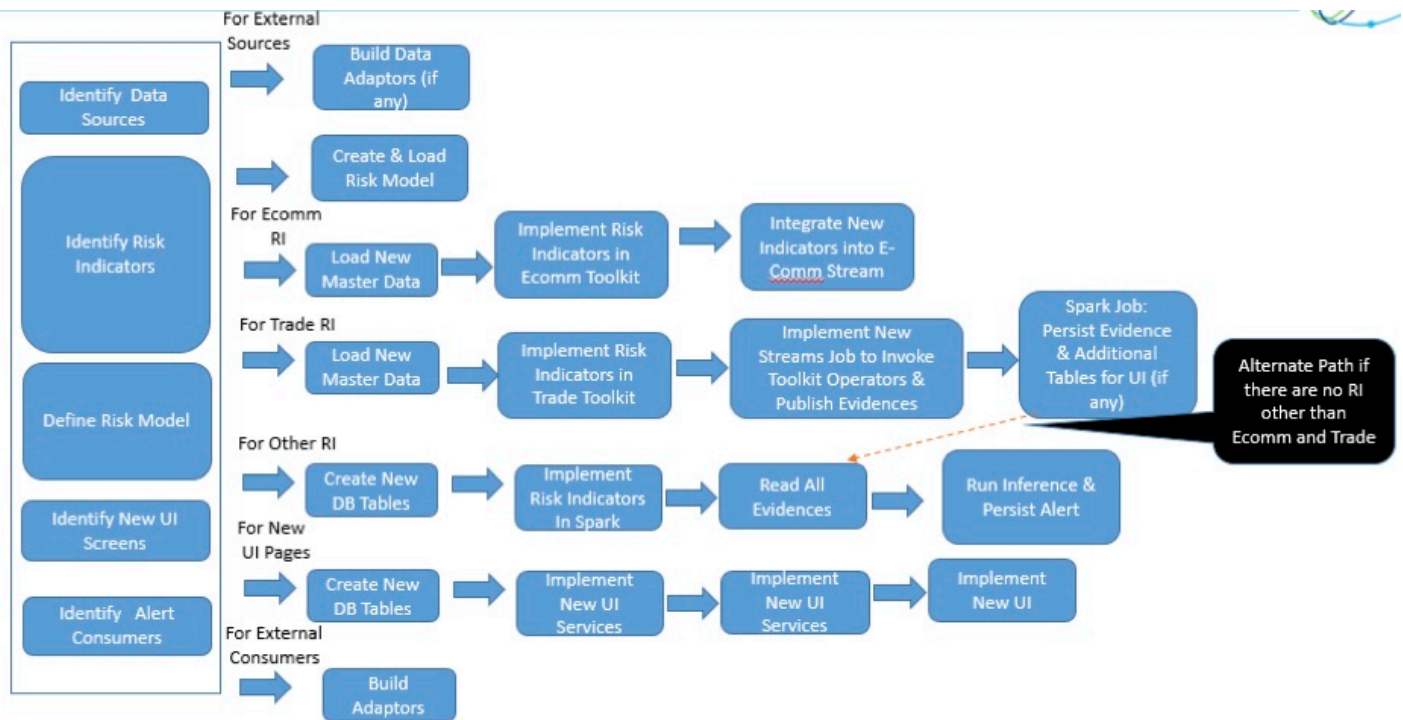


Figure 46: Tasks for developing your own use cases

1. Based on the external sources of data (such as e-comm and trade), you might have to build adaptors to get the data from customer sources systems to a format that Surveillance Insight can process.
2. Irrespective of the type of use case (trade or e-comm), you must create a risk model and load it into the Surveillance Insight database.
3. A different series of steps must be followed to develop e-comm and trade risk indicators.
4. Risk Indicators that do not fall under either the e-comm or trade categories can be implemented in Spark as a job that runs at the end of the day. This job can be used to collect all of the evidences (from trade and e-comm) and run the inference engine. If the "Other RI" path in the diagram is not required for a specific use case, the collection of evidences and invocation of the inference engine can be done from the Spark job for Trade Surveillance.
5. During implementation, ensure that the e-comm, trade, and other RI evidences are persisted before you run the inference engine. When the inference engine is run (either intra-day or at the end of the day) depends on the requirements of your use case.
6. Some use cases require additional charts to show details of the evidences beyond what is shown by the default in the Surveillance Insight Workbench. Such pages might require more tables in the Surveillance Insight database, more REST services to serve the UI, and the UI pages to be developed based on the requirements.
7. Alerts that are generated can be passed on to other consumer systems such as case managers by building custom adaptors.

Requirement analysis

Before you begin developing your IBM Surveillance Insight for Financial Services use case, you must analyze the requirements.

After you define the business requirements, the next step is to translate the requirements in terms that are relevant to Surveillance Insight:

1. Identify the risk indicators that are relevant to the use case.
2. Classify the risk indicators under the trade and/or e-comms categories.
3. Define the relationships between the indicators. For example, some indicators might depend on other indicators. Such dependencies help to decide the sequencing of the risk evidences and the technology that can be used to implement the risk indicators, such as either Streams or Spark.
4. Identify the alert(s) that might have to be generated for the specific use case.
5. For each alert, identify when the alert needs to be generated in terms of evidences, how long the alert is expected to span in time, and the entities related to the alert, such as tickers, parties, and communications.
6. Define the user interface screens that might be required and the default alert screens that are part of the Surveillance Insight Workbench.

Data design

Regarding the input data, you must do the following steps:

1. Identify the external data (such as stock market data, email data, voice data) that is associated with each risk indicator. This should be clear from the definition of each risk indicator in the requirement analysis step.
2. Identify the data that is associated with any additional user interfaces that might be identified in the requirement analysis step.
3. Identify the sources for the data from step 1 and 2 above. This may help in identifying adaptors that might be needed to read the data from external sources.
4. Create a list of master data entries that are required for the RISK_INDICATOR_MASTER and ALERT_TYPE master tables.

Define the risk model

Based on the risk indicators and inference logic for the use case, you must define the risk model network for the use case. For more information about risk models, see [Chapter 7, “Inference engine,” on page 71](#).

Risk indicators and data types

Before you can design a new risk indicator, review the contents of the Trade Surveillance Toolkit and the E-Comm Surveillance Toolkit to see if there are existing operators that you can reuse.

For more information, see [“Trade Surveillance Toolkit” on page 25](#) and [“E-Comm Surveillance Toolkit” on page 35](#).

The following considerations must be made for each risk indicator:

- Identify what logic each indicator needs to implement and how the result translated into a risk score.
- Identify what data needs to be provided to downstream components to be able to create the right user interface screens.
- Identify the right place in the architecture to implement the indicator. Some indicators are best developed in Streams, while others are best developed in Spark.

Other design considerations

Identify Kafka messaging topics that are required for integration.

At a minimum, one topic is required to communicate between the trade Streams and trade Spark jobs to publishing the risk evidence. A similar topic for e-comms evidence is already part of the Surveillance Insight installation.

A Kafka topic for incoming e-comms and voice data is also part of the Surveillance Insight installation.

Implementation

At a high level, implementation involves the following steps:

- Load master data
- Implement risk indicators
- Persist evidences
- Invoke inference engine

- Create and/or update an alert
- Implement more services that might be required for the additional user interfaces (if any)
- Implement and integrate more user interfaces (if any)

For more information, see [“Extending Trade Surveillance” on page 31](#) and [“Extending E-Comm Surveillance” on page 53](#).

Chapter 11. Troubleshooting

This section provides troubleshooting information.

Solution installer is unable to create the chef user

You are installing the components by using the solution installer. When you run the `setup.sh` script, you receive an error message that the solution installer is "Unable to create the chef user".

You can resolve this problem, by running the `cleanup.sh` script and then restarting the computer.

NoSuchAlgorithmException

While you are running the pump and dump Spark job, you receive the following error:

```
INFO ApplicationMaster: Final app status: FAILED, exitCode: 15, (reason: User class
  threw exception:org.apache.spark.SparkException: Job aborted due to stage
  failure:Task 0 in stage 6.0
    failed 4 times, most recent failure: Lost task 0.3 in stage 6.0 (TID 9,
  civcmk339.in.ibm.com):
    com.ibm.security.pkcsutil.PKCSException: Content decryption error
  (java.security.NoSuchAlgorithmException:
    AES SecretKeyFactory not available)
  at com.ibm.security.pkcs7.EncryptedContentInfo.decrypt(EncryptedContentInfo.java:1019)
  at com.ibm.security.pkcs7.EnvelopedData.decrypt(EnvelopedData.java:1190)
  at com.ibm.si.security.util.SISecurityUtil.decrypt(SISecurityUtil.java:73)
  at com.ibm.sifs.evidence.PnDCollectEvidence$2$1.call(PnDCollectEvidence.java:122)
  at com.ibm.sifs.evidence.PnDCollectEvidence$2$1.call(PnDCollectEvidence.java:105)
  at org.apache.spark.api.java.JavaPairRDD$$anonfun$toScalaFunction
  $1.apply(JavaPairRDD.scala:1028)
  at scala.collection.Iterator$$anon$11.next(Iterator.scala:409)
  at scala.collection.Iterator$$anon$12.nextCur(Iterator.scala:434)
  at scala.collection.Iterator$$anon$12.hasNext(Iterator.scala:440)
  at scala.collection.Iterator$class.foreach(Iterator.scala:893)
  at scala.collection.AbstractIterator.foreach(Iterator.scala:1336)
  at scala.collection.TraversableOnce$class.foldLeft(TraversableOnce.scala:157)
  at scala.collection.AbstractIterator.foldLeft(Iterator.scala:1336)
  at scala.collection.TraversableOnce$class.fold(TraversableOnce.scala:212)
  at scala.collection.AbstractIterator.fold(Iterator.scala:1336)
  at org.apache.spark.rdd.RDD$$anonfun$fold$1$$anonfun$20.apply(RDD.scala:1063)
  at org.apache.spark.rdd.RDD$$anonfun$fold$1$$anonfun$20.apply(RDD.scala:1063)
  at org.apache.spark.SparkContext$$anonfun$32.apply(SparkContext.scala:1935)
  at org.apache.spark.SparkContext$$anonfun$32.apply(SparkContext.scala:1935)
  at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:70)
  at org.apache.spark.scheduler.Task.run(Task.scala:86)
  at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:274)
  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
  at java.lang.Thread.run(Thread.java:745)
```

To resolve this problem, ensure that the algorithm and keylength values are correctly set for the IBM Streams `encrypt.properties` file.

1. Go to the `/home/streamsadmin/config/properties` directory.
2. Open `encrypt.properties` in a text editor.
3. Ensure that the following values are set:
 - `algorithm=3DES`

- keylength=168

java.lang.ClassNotFoundException: com.ibm.security.pkcsutil.PKCSException

While you are running the pump and dump Spark job, you receive the following error:

```
java.lang.ClassNotFoundException: com.ibm.security.pkcsutil.PKCSException
at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
at java.lang.Class.forName0(Native Method)
at java.lang.Class.forName(Class.java:348)
at org.apache.spark.serializer.JavaDeserializationStream$$anon$1.resolveClass(JavaSerializer.scala:67)
at java.io.ObjectInputStream.readNonProxyDesc(ObjectInputStream.java:1613)
at java.io.ObjectInputStream.readClassDesc(ObjectInputStream.java:1518)
at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1774)
at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1351)
at java.io.ObjectInputStream.readObject(ObjectInputStream.java:371)
at org.apache.spark.ThrowableSerializationWrapper.readObject(TaskEndReason.scala:183)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at java.io.ObjectStreamClass.invokeReadObject(ObjectStreamClass.java:1058)
at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1900)
at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1801)
at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1351)
at java.io.ObjectInputStream.defaultReadFields(ObjectInputStream.java:2000)
at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1924)
at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1801)
at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1351)
at java.io.ObjectInputStream.defaultReadFields(ObjectInputStream.java:2000)
at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1924)
at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1801)
at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1351)
at java.io.ObjectInputStream.readObject(ObjectInputStream.java:371)
at org.apache.spark.serializer.JavaDeserializationStream.readObject(JavaSerializer.scala:75)
at org.apache.spark.serializer.JavaSerializerInstance.deserialize(JavaSerializer.scala:114)
at org.apache.spark.scheduler.TaskResultGetter$$anon$3$$anonfun$run$2.apply$mcV$sp(TaskResultGetter.scala:128)
at org.apache.spark.scheduler.TaskResultGetter$$anon$3$$anonfun$run$2.apply(TaskResultGetter.scala:124)
at org.apache.spark.scheduler.TaskResultGetter$$anon$3$$anonfun$run$2.apply(TaskResultGetter.scala:124)
at org.apache.spark.util.Utils$.logUncaughtExceptions(Utils.scala:1953)
at org.apache.spark.scheduler.TaskResultGetter$$anon$3.run(TaskResultGetter.scala:124)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
```

To resolve this problem, ensure that the `ibmpkcs.jar` file is present in the `--conf="spark.driver.extraClassPath=` value when you run the PartyRiskScoring Spark job. For more information, see [Deploying PartyRiskScoring in the installation guide \(https://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/install/t_trd_deploypartyriskscoringspark.html\)](https://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/install/t_trd_deploypartyriskscoringspark.html).

java.lang.NoClassDefFoundError: com/ibm/security/pkcs7/Content

While you are running the pump and dump Spark job, you receive the following error:

```
17/01/20 15:22:21 WARN TaskSetManager: Lost task 0.0 in stage 1.0 (TID 1,
  civcmk339.in.ibm.com): java.lang.NoClassDefFoundError: com/ibm/security/pkcs7/Content
    at com.ibm.sifs.evidence.PnDCollectEvidence
$2$1.call(PnDCollectEvidence.java:122)
    at com.ibm.sifs.evidence.PnDCollectEvidence
$2$1.call(PnDCollectEvidence.java:105)
    at org.apache.spark.api.java.JavaPairRDD$$anonfun$toScalaFunction
$1.apply(JavaPairRDD.scala:1028)
    at scala.collection.Iterator$$anon$11.next(Iterator.scala:409)
    at scala.collection.Iterator$$anon$12.nextCur(Iterator.scala:434)
    at scala.collection.Iterator$$anon$12.hasNext(Iterator.scala:440)
    at scala.collection.Iterator$class.foreach(Iterator.scala:893)
    at scala.collection.AbstractIterator.foreach(Iterator.scala:1336)
    at scala.collection.TraversableOnce$class.foldLeft(TraversableOnce.scala:157)
    at scala.collection.AbstractIterator.foldLeft(Iterator.scala:1336)
    at scala.collection.TraversableOnce$class.fold(TraversableOnce.scala:212)
    at scala.collection.AbstractIterator.fold(Iterator.scala:1336)
    at org.apache.spark.rdd.RDD$$anonfun$fold$1$$anonfun$20.apply(RDD.scala:1063)
    at org.apache.spark.rdd.RDD$$anonfun$fold$1$$anonfun$20.apply(RDD.scala:1063)
    at org.apache.spark.SparkContext$$anonfun$32.apply(SparkContext.scala:1935)
    at org.apache.spark.SparkContext$$anonfun$32.apply(SparkContext.scala:1935)
    at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:70)
    at org.apache.spark.scheduler.Task.run(Task.scala:86)
    at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:274)
    at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
    at java.util.concurrent.ThreadPoolExecutor
$Worker.run(ThreadPoolExecutor.java:617)
    at java.lang.Thread.run(Thread.java:745)
Caused by: java.lang.ClassNotFoundException: com.ibm.security.pkcs7.Content
    at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:331)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    ... 22 more
```

To resolve this problem, ensure that the `ibmpkcs.jar` file is present in the `--conf="spark.driver.extraClassPath=` value when you run the PartyRiskScoring Spark job. For more information, see [Deploying PartyRiskScoring in the installation guide](https://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/install/t_trd_deploypartyriskscoringspark.html) (https://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/install/t_trd_deploypartyriskscoringspark.html).

java.lang.NoClassDefFoundError: org/apache/spark/streaming/kafka010/LocationStrategies

If you receive this error, ensure that `spark-streaming_2.11-2.0.1.jar` is in the `jars` folder where you installed Spark.

java.lang.NoClassDefFoundError (com/ibm/si/security/util/SISecurityUtil)

While you are running the spoofing Spark job, you receive the following error:

```
17/04/25 18:51:41 INFO TaskSetManager: Lost task 0.3 in stage 222.0 (TID 225) on
executor
  civcmk339.in.ibm.com: java.lang.NoClassDefFoundError (com/ibm/si/security/util/
SISecurityUtil) [duplicate 3]
```

```

17/04/25 18:51:41 ERROR TaskSetManager: Task 0 in stage 222.0 failed 4 times; aborting
job
17/04/25 18:51:41 INFO YarnClusterScheduler: Removed TaskSet 222.0, whose tasks have
all completed, from pool
17/04/25 18:51:41 INFO YarnClusterScheduler: Cancelling stage 222
17/04/25 18:51:41 INFO DAGScheduler: ResultStage 222 (next at
SpoofingEvidence.java:216) failed in 1.272 s
17/04/25 18:51:41 INFO DAGScheduler: Job 222 failed: next at
SpoofingEvidence.java:216, took 1.279829 s
17/04/25 18:51:41 INFO JobScheduler: Finished job streaming job 1493126500000 ms.0
from job set of time 1493126500000 ms
17/04/25 18:51:41 INFO JobScheduler: Total delay: 1.746 s for time 1493126500000 ms
(execution: 1.739 s)
17/04/25 18:51:41 INFO MapPartitionsRDD: Removing RDD 441 from persistence list
17/04/25 18:51:41 INFO KafkaRDD: Removing RDD 440 from persistence list
17/04/25 18:51:41 INFO BlockManager: Removing RDD 440
17/04/25 18:51:41 INFO ReceivedBlockTracker: Deleting batches:
17/04/25 18:51:41 INFO BlockManager: Removing RDD 441
17/04/25 18:51:41 INFO InputInfoTracker: remove old batch metadata: 1493126480000 ms
17/04/25 18:51:41 ERROR JobScheduler: Error running job streaming job 1493126500000
ms.0
org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage
222.0 failed 4 times, most recent failure:
  Lost task 0.3 in stage 222.0 (TID 225, civcmk339.in.ibm.com):
java.lang.NoClassDefFoundError: com/ibm/sifs/security/util/SISecurityUtil
    at com.ibm.sifs.evidence.SpoofingEvidence$5.call(SpoofingEvidence.java:372)
    at com.ibm.sifs.evidence.SpoofingEvidence$5.call(SpoofingEvidence.java:348)
    at org.apache.spark.api.java.JavaPairRDD$$anonfun$toScalaFunction
$1.apply(JavaPairRDD.scala:1028)
        at scala.collection.Iterator$$anon$11.next(Iterator.scala:409)
        at scala.collection.Iterator$class.foreach(Iterator.scala:893)
        at scala.collection.AbstractIterator.foreach(Iterator.scala:1336)
        at scala.collection.generic.Growable$class.$plus$plus$eq(Growable.scala:59)
        at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:104)
        at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:48)
        at scala.collection.TraversableOnce$class.to(TraversableOnce.scala:310)
        at scala.collection.AbstractIterator.to(Iterator.scala:1336)
        at scala.collection.TraversableOnce$class.toBuffer(TraversableOnce.scala:302)
        at scala.collection.AbstractIterator.toBuffer(Iterator.scala:1336)
        at scala.collection.TraversableOnce$class.toArray(TraversableOnce.scala:289)
        at scala.collection.AbstractIterator.toArray(Iterator.scala:1336)
        at org.apache.spark.rdd.RDD$$anonfun$toLocalIterator$1$$anonfun$org$apache
$spark$rdd$RDD$$anonfun$$collectPartition$1$1.apply(RDD.scala:927)
            at org.apache.spark.rdd.RDD$$anonfun$toLocalIterator$1$$anonfun$org$apache
$spark$rdd$RDD$$anonfun$$collectPartition$1$1.apply(RDD.scala:927)
            at org.apache.spark.SparkContext$$anonfun$runJob
$5.apply(SparkContext.scala:1899)
                at org.apache.spark.SparkContext$$anonfun$runJob
$5.apply(SparkContext.scala:1899)
                    at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:70)
                    at org.apache.spark.scheduler.Task.run(Task.scala:86)
                    at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:274)
                    at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
        at java.util.concurrent.ThreadPoolExecutor
$Worker.run(ThreadPoolExecutor.java:617)
        at java.lang.Thread.run(Thread.java:745)

```

To resolve this problem, ensure the following:

- The jar versions are correct in the Spark submit command.
- The jar files are in the /home/sifsuser/lib on all of the IBM Open Platform nodes.

org.json.JSONException:A JSONObject text must begin with '{' at character 1

While you are running the spoofing Spark job, you receive the following error:

```
org.json.JSONException: A JSONObject text must begin with '{' at character 1
    at org.json.JSONTokener.syntaxError(JSONTokener.java:410)
    at org.json.JSONObject.<init>(JSONObject.java:179)
    at org.json.JSONObject.<init>(JSONObject.java:402)
    at
com.ibm.sifs.evidence.DB.EvidenceDBUpdate.createEvidence(EvidenceDBUpdate.java:238)
    at
com.ibm.sifs.evidence.SpoofingEvidence.createSpoofingEvidence(SpoofingEvidence.java:416)
)
    at com.ibm.sifs.evidence.SpoofingEvidence.access$200(SpoofingEvidence.java:90)
    at com.ibm.sifs.evidence.SpoofingEvidence$2.call(SpoofingEvidence.java:230)
    at com.ibm.sifs.evidence.SpoofingEvidence$2.call(SpoofingEvidence.java:194)
    at org.apache.spark.streaming.api.java.JavaDStreamLike$$anonfun$foreachRDD
$1.apply(JavaDStreamLike.scala:272)
    at org.apache.spark.streaming.api.java.JavaDStreamLike$$anonfun$foreachRDD
$1.apply(JavaDStreamLike.scala:272)
    at org.apache.spark.streaming.dstream.DStream$$anonfun$foreachRDD$1$$anonfun
$apply$mcV$sp$3.apply(DStream.scala:627)
    at org.apache.spark.streaming.dstream.DStream$$anonfun$foreachRDD$1$$anonfun
$apply$mcV$sp$3.apply(DStream.scala:627)
    at org.apache.spark.streaming.dstream.ForEachDStream$$anonfun$1$$anonfun$apply
$mcV$sp$1.apply$mcV$sp(ForEachDStream.scala:51)
    at org.apache.spark.streaming.dstream.ForEachDStream$$anonfun$1$$anonfun$apply
$mcV$sp$1.apply(ForEachDStream.scala:51)
    at org.apache.spark.streaming.dstream.ForEachDStream$$anonfun$1$$anonfun$apply
$mcV$sp$1.apply(ForEachDStream.scala:51)
    at
org.apache.spark.streaming.dstream.DStream.createRDDWithLocalProperties(DStream.scala:4
15)
    at org.apache.spark.streaming.dstream.ForEachDStream$$anonfun$1.apply$mcV
$sp(ForEachDStream.scala:50)
    at org.apache.spark.streaming.dstream.ForEachDStream$$anonfun
$1.apply(ForEachDStream.scala:50)
    at org.apache.spark.streaming.dstream.ForEachDStream$$anonfun
$1.apply(ForEachDStream.scala:50)
    at scala.util.Try$.apply(Try.scala:192)
    at org.apache.spark.streaming.scheduler.Job.run(Job.scala:39)
    at org.apache.spark.streaming.scheduler.JobScheduler$JobHandler$$anonfun$run
$1.apply$mcV$sp(JobScheduler.scala:247)
    at org.apache.spark.streaming.scheduler.JobScheduler$JobHandler$$anonfun$run
$1.apply(JobScheduler.scala:247)
    at org.apache.spark.streaming.scheduler.JobScheduler$JobHandler$$anonfun$run
$1.apply(JobScheduler.scala:247)
    at scala.util.DynamicVariable.withValue(DynamicVariable.scala:58)
    at org.apache.spark.streaming.scheduler.JobScheduler
$JobHandler.run(JobScheduler.scala:246)
    at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
    at java.util.concurrent.ThreadPoolExecutor
$Worker.run(ThreadPoolExecutor.java:617)
    at java.lang.Thread.run(Thread.java:745)
```

Ensure that the trader IDs [SAL2493, KAP0844] are present in the sifs.party_master table.

javax.servlet.ServletException:Could not find endpoint information

While you are running the e-comm Spark job, you receive the following error:

```
[4/26/17 12:29:03:324 IST] 0000006d ServletWrape E
com.ibm.ws.webcontainer.servlet.ServletWrapper
  init SRVE0271E: Uncaught init() exception created by servlet [SIDataRESTApp] in
application
  [com_ibm_sifs_service_data-2_0_0-SNAPSHOT_war]: javax.servlet.ServletException:
Coult not find endpoint information.
at com.ibm.websphere.jaxrs.server.IBMRestServlet.init(IBMRestServlet.java:87)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.init(ServletWrapper.java:342)
at com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.init(ServletWrapperImpl.java:168)
at
com.ibm.ws.webcontainer.servlet.ServletWrapper.loadOnStartupCheck(ServletWrapper.java:1
376)
at com.ibm.ws.webcontainer.webapp.WebApp.doLoadOnStartupActions(WebApp.java:662)
at com.ibm.ws.webcontainer.webapp.WebApp.commonInitializationFinally(WebApp.java:628)
at com.ibm.ws.webcontainer.webapp.WebAppImpl.initialize(WebAppImpl.java:453)
at com.ibm.ws.webcontainer.webapp.WebGroupImpl.addWebApplication(WebGroupImpl.java:88)
at com.ibm.ws.webcontainer.VirtualHostImpl.addWebApplication(VirtualHostImpl.java:171)
at com.ibm.ws.webcontainer.WSWebContainer.addWebApp(WSWebContainer.java:904)
at com.ibm.ws.webcontainer.WSWebContainer.addWebApplication(WSWebContainer.java:789)
at
com.ibm.ws.webcontainer.component.WebContainerImpl.install(WebContainerImpl.java:427)
at com.ibm.ws.webcontainer.component.WebContainerImpl.start(WebContainerImpl.java:719)
at com.ibm.ws.runtime.component.ApplicationMgrImpl.start(ApplicationMgrImpl.java:1246)
at
com.ibm.ws.runtime.component.DeployedApplicationImpl.fireDeployedObjectStart(DeployedAp
plicationImpl.java:1514)
at com.ibm.ws.runtime.component.DeployedModuleImpl.start(DeployedModuleImpl.java:704)
at
com.ibm.ws.runtime.component.DeployedApplicationImpl.start(DeployedApplicationImpl.java
:1096)
at
com.ibm.ws.runtime.component.ApplicationMgrImpl.startApplication(ApplicationMgrImpl.jav
a:798)
at com.ibm.ws.runtime.component.ApplicationMgrImpl$5.run(ApplicationMgrImpl.java:2314)
at com.ibm.ws.security.auth.ContextManagerImpl.runAs(ContextManagerImpl.java:5489)
at
com.ibm.ws.security.auth.ContextManagerImpl.runAsSystem(ContextManagerImpl.java:5615)
at com.ibm.ws.security.core.SecurityContext.runAsSystem(SecurityContext.java:255)
at com.ibm.ws.runtime.component.ApplicationMgrImpl.start(ApplicationMgrImpl.java:2319)
at
com.ibm.ws.runtime.component.CompositionUnitMgrImpl.start(CompositionUnitMgrImpl.java:4
36)
at com.ibm.ws.runtime.component.CompositionUnitImpl.start(CompositionUnitImpl.java:123)
at
com.ibm.ws.runtime.component.CompositionUnitMgrImpl.start(CompositionUnitMgrImpl.java:3
79)
at com.ibm.ws.runtime.component.CompositionUnitMgrImpl.access
$500(CompositionUnitMgrImpl.java:127)
at com.ibm.ws.runtime.component.CompositionUnitMgrImpl
$CUInitializer.run(CompositionUnitMgrImpl.java:985)
at com.ibm.wsspi.runtime.component.WsComponentImpl
$_AsynchInitializer.run(WsComponentImpl.java:524)
at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1892)
```

To resolve this problem, verify your JAX RS settings in WebSphere Application Server.

1. Log into the WebSphere Application Server admin console.
2. Change the JAX RS version settings:
 - a. Expand **Servers** > **Server Types** > **WebSphere Application Servers** and then click the server.

- b. Click **Container Services** and then click **Default JAXRS provider settings**.
- c. Change the **JAX RS Provider** setting to 1.1.

DB2INST1.COMM_POLICY is an undefined name

While you are running the e-comm curl command, you receive the following error:

```
"status": "500", "message": "\"DB2INST1.COMM_POLICY\" is an undefined name..  
SQLCODE=-204, SQLSTATE=42704, DRIVER=4.22.29
```

To resolve this problem, ensure that the currentSchema is set to SIFS in the WebSphere Application Server custom properties.

Authorization Error 401

While you are running the e-comm curl command, you receive an Authorization Error 401.

To resolve this problem, ensure that the com.ibm.sifs.security-2.0.0-SNAPSHOT.jar security jar file is in the lib/ext folder for WebSphere Application Server.

javax.security.sasl.SaslException: GSS initiate failed

While performing some Hadoop file system operations, you receive the following error:

```
17/05/09 23:00:21 INFO client.RMProxy: Connecting to ResourceManager at  
civcmk337.in.ibm.com/9.194.241.124:8050  
17/05/09 23:00:21 WARN ipc.Client: Exception encountered while connecting to the  
server : javax.security.sasl.SaslException: GSS  
initiate failed [Caused by GSSException: No valid credentials provided (Mechanism  
level: Failed to find any Kerberos tgt)]  
Exception in thread "main" java.io.IOException: Failed on local exception:  
java.io.IOException: javax.security.sasl.SaslException:  
GSS initiate failed [Caused by GSSException: No valid credentials provided  
(Mechanism level: Failed to find any Kerberos tgt)]; Host Details  
: local host is: "civcmk336.in.ibm.com/9.194.241.123"; destination host is:  
"civcmk337.in.ibm.com":8050;  
at org.apache.hadoop.net.NetUtils.wrapException(NetUtils.java:776)  
at org.apache.hadoop.ipc.Client.call(Client.java:1479)  
at org.apache.hadoop.ipc.Client.call(Client.java:1412)  
at org.apache.hadoop.ipc.ProtobufRpcEngine$Invoker.invoke(ProtobufRpcEngine.java:229)  
at com.sun.proxy.$Proxy7.getApplications(Unknown Source)  
at  
org.apache.hadoop.yarn.api.impl.pb.client.ApplicationClientProtocolPBClientImpl.getApplica  
tions(ApplicationClientProtocolPBClientImpl.java:251)  
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)  
at  
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  
at java.lang.reflect.Method.invoke(Method.java:498)  
at  
org.apache.hadoop.io.retry.RetryInvocationHandler.invokeMethod(RetryInvocationHandler.j  
ava:191)  
at  
org.apache.hadoop.io.retry.RetryInvocationHandler.invoke(RetryInvocationHandler.java:10  
2)  
at com.sun.proxy.$Proxy8.getApplications(Unknown Source)  
at  
org.apache.hadoop.yarn.client.api.impl.YarnClientImpl.getApplications(YarnClientImpl.ja  
va:478)  
at
```

```
org.apache.hadoop.yarn.client.cli.ApplicationCLI.listApplications(ApplicationCLI.java:401)
at org.apache.hadoop.yarn.client.cli.ApplicationCLI.run(ApplicationCLI.java:207)
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:84)
at org.apache.hadoop.yarn.client.cli.ApplicationCLI.main(ApplicationCLI.java:83)
Caused by: java.io.IOException: javax.security.sasl.SaslException: GSS initiate failed
[Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
at org.apache.hadoop.ipc.Client$Connection$1.run(Client.java:687)
at java.security.AccessController.doPrivileged(Native Method)
at javax.security.auth.Subject.doAs(Subject.java:422)
at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1657)
at org.apache.hadoop.ipc.Client$Connection.handleSaslConnectionFailure(Client.java:650)
at org.apache.hadoop.ipc.Client$Connection.setupIOstreams(Client.java:737)
at org.apache.hadoop.ipc.Client$Connection.access$2900(Client.java:375)
at org.apache.hadoop.ipc.Client.getConnection(Client.java:1528)
at org.apache.hadoop.ipc.Client.call(Client.java:1451)
... 17 more
```

You can resolve this problem, run the `knit` command with the appropriate user and keytab. For example, `kinit -kt /etc/security/keytabs/sifsuser.keytab sifsuser@IBM.COM`.

PacketFileSource_1_out0.so file....:undefined symbol:libusb_open

While you are running the voice Streams job, you receive the following error:

```
An error occurred while the dynamic shared object was loaded for the processing
element from the /home/streamsadmin/.streams/var/
Streams.sab_ALZ-StreamsDomain-SIIE/9d3184d2-9701-4d7c-84dc-b641a18effb7/60033b3.../
setup/bin/PacketFileSource_1_out0.so
file....:undefined symbol:libusb_open.so - Voice Streams job
```

To resolve this problem, install the `libpcap-devel` RPM on your Linux operating system.

[Servlet Error]-[SIFSRestService]: java.lang.IllegalArgumentException

If you receive this error, ensure that the JNDI variables in the WebSphere Application Server `SOLR_PROXY_URL` and `SOLR_SERVER_URL` are set correctly.

Failed to update metadata after 60000 ms

If you receive this error while running the `processVoice.sh` script, ensure that the `bootstrap.servers` value in the `kafka.properties` file is set to a valid IP address, and not set to `localhost`.

java.lang.NoClassDefFoundError: com/sun/jersey/api/client/config/ClientConfig

If you receive this error while running the pump and dump Spark job, ensure that you have cleared the `yarn-timeline-service.enabled` option in the Ambari console.

For more information, see the [Installing Apache Spark](https://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/install/t_trd_installingspark.html) in the installation guide (https://www.ibm.com/support/knowledgecenter/SSWTQQ_2.0.0/docs/install/t_trd_installingspark.html).

DB2 SQL Error: SQLCODE=-204, SQLSTATE=42704, SQLERRMC=DB2INST1.COMM_TYPE_MASTER

If you receive this message when you run the e-comm Spark job, ensure that the jdbcurl value set in the `sifs.spark.properties` file has "currentschema=SIFS" set at the end. For example, `jdbc:db2://<IP>:50001/SIFS:sslConnection=true;currentSchema=SIFS`;

org.apache.kafka.common.config.ConfigException: Invalid url in bootstrap.servers

While you are running the e-comm Streams job, you receive the following error:

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Exception in thread "Thread-29" org.apache.kafka.common.KafkaException: Failed to
construct kafka consumer
    at org.apache.kafka.clients.consumer.KafkaConsumer.<init>(KafkaConsumer.java:702)
    at org.apache.kafka.clients.consumer.KafkaConsumer.<init>(KafkaConsumer.java:587)
    at org.apache.kafka.clients.consumer.KafkaConsumer.<init>(KafkaConsumer.java:569)
    at
com.ibm.streamsx.messaging.kafka.KafkaConsumerClient.<init>(KafkaConsumerClient.java:45
)
    at
com.ibm.streamsx.messaging.kafka.KafkaByteArrayConsumerV9.<init>(KafkaByteArrayConsumer
V9.java:16)
    at
com.ibm.streamsx.messaging.kafka.KafkaConsumerFactory.getClient(KafkaConsumerFactory.ja
va:47)
    at
com.ibm.streamsx.messaging.kafka.KafkaSource.getNewConsumerClient(KafkaSource.java:132)
    at com.ibm.streamsx.messaging.kafka.KafkaSource.initialize(KafkaSource.java:115)
    at
com.ibm.streamsx.operator.internal.runtime.api.OperatorAdapter.initialize(OperatorAdapte
r.java:735)
    at com.ibm.streamsx.operator.internal.jni.JNIBridge.<init>(JNIBridge.java:271)
Caused by: org.apache.kafka.common.config.ConfigException: Invalid url in
bootstrap.servers: <IP>:<Port>
    at org.apache.kafka.clients.ClientUtils.parseAndValidateAddresses(ClientUtils.java:45)
    at org.apache.kafka.clients.consumer.KafkaConsumer.<init>(KafkaConsumer.java:620)
```

To resolve this problem, ensure that the `bootstrap.servers` value in `/home/streamsadmin/config/properties/consumer.properties` is set to correct IP address and port number.

[localhost/127.0.0.1, localhost/0:0:0:0:0:0:1] failed: Connection refused

While you are running the e-comm Streams job, you receive the following error:

```
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Exception in thread "Thread-84" org.apache.http.conn.HttpHostConnectException: Connect
to localhost:9443 [localhost/127.0.0.1,
localhost/0:0:0:0:0:0:1] failed: Connection refused
    at
org.apache.http.impl.conn.DefaultHttpClientConnectionOperator.connect(DefaultHttpClient
ConnectionOperator.java:158)
    at
org.apache.http.impl.conn.PoolingHttpClientConnectionManager.connect(PoolingHttpClientC
onnectionManager.java:353)
    at
```

```

org.apache.http.impl.execchain.MainClientExec.execute(MainClientExec.java:380)
    at
org.apache.http.impl.execchain.MainClientExec.execute(MainClientExec.java:236)
    at org.apache.http.impl.execchain.ProtocolExec.execute(ProtocolExec.java:184)
    at org.apache.http.impl.execchain.RetryExec.execute(RetryExec.java:88)
    at org.apache.http.impl.execchain.RedirectExec.execute(RedirectExec.java:110)
    at
org.apache.http.impl.client.InternalHttpClient.doExecute(InternalHttpClient.java:184)
    at
org.apache.http.impl.client.CloseableHttpClient.execute(CloseableHttpClient.java:82)
    at com.ibm.sifs.ecomm.Utility.invokeService(Utility.java:77)
    at com.ibm.sifs.ecomm.PolicyLoader.loadAllPolicies(PolicyLoader.java:211)
    at com.ibm.sifs.ecomm.PolicyLoader.process(PolicyLoader.java:157)

```

To resolve this problem, ensure that the POLICYRESTURL value is set to the IP address and port number of the WebSphere Application Server where the CommServices is. For example, POLICYRESTURL=https://localhost:9443/CommServices/ecommerce/policy/.

java.lang.Exception: 500 response code received from REST service

While you are running the e-comm Streams job, you receive the following error:

```

SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Exception in thread "Thread-85" java.lang.Exception: 500 response code received from
REST service!!
at com.ibm.sifs.ecomm.Utility.invokeService(Utility.java:101)
at com.ibm.sifs.ecomm.PolicyLoader.loadAllPolicies(PolicyLoader.java:211)
at com.ibm.sifs.ecomm.PolicyLoader.process(PolicyLoader.java:157)
at
com.ibm.streams.operator.internal.runtime.api.OperatorAdapter.processTuple(OperatorAdap
ter.java:1591)
at com.ibm.streams.operator.internal.runtime.api.OperatorAdapter
$1.tuple(OperatorAdapter.java:825)
at com.ibm.streams.operator.internal.runtime.api.OperatorAdapter
$1.tuple(OperatorAdapter.java:818)
at com.ibm.streams.operator.internal.ports.SwitchingHandler
$Alternate.tuple(SwitchingHandler.java:162)
at
com.ibm.streams.operator.internal.network.DeserializingStream.tuple(DeserializingStream
.java:76)
at
com.ibm.streams.operator.internal.network.DeserializingStream.processRawTuple(Deseriali
zingStream.java:65)
at
com.ibm.streams.operator.internal.runtime.api.InputPortsConduit.processRawTuple(InputPo
rtsConduit.java:100)
at com.ibm.streams.operator.internal.jni.JNIBridge.process(JNIBridge.java:312)
Couldn't open cc.mallet.util.MalletLogger resources/logging.properties file.
Perhaps the 'resources' directories weren't copied into the 'class' directory.

```

To resolve this problem, ensure the following:

- The JNDI variable ACTIANCE_USERS is set in WebSphere Application Server and is valid.
- The users.json file on the Streams node contains valid user details.
- The ibmrest1 user that you created in WebSphere Application Server as part of the Managed users should have the same credentials that were used in the users.json.
- com_ibm-sifs_service_data-2_0_0-SNAPSHOT.war was deployed in detailed mode in WebSphere Application Server and the users are mapped correctly.
- The ibmrest1 user is part of the admin group.

javax.security.auth.login.FailedLoginException: Null

While you are running the pump-and-dump or spoofing Streams jobs, you receive the following error:

```
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Exception in thread "Thread-11" java.io.IOException: Login failure for
sifsuser@IBM.COM from keytab
/etc/security/keytabs/sifsuser.keytab
at
org.apache.hadoop.security.UserGroupInformation.loginUserFromKeytabAndReturnUGI(UserGro
upInformation.java:1146)
at
com.ibm.streamsx.hdfs.client.auth.BaseAuthenticationHelper.authenticateWithKerberos(Bas
eAuthenticationHelper.java:104)
at
com.ibm.streamsx.hdfs.client.auth.HDFSAuthenticationHelper.connect(HDFSAuthenticationHe
lper.java:59)
at com.ibm.streamsx.hdfs.client.AbstractHdfsClient.connect(AbstractHdfsClient.java:35)
at com.ibm.streamsx.hdfs.client.HdfsJavaClient.connect(HdfsJavaClient.java:10)
at com.ibm.streamsx.hdfs.AbstractHdfsOperator.initialize(AbstractHdfsOperator.java:56)
at com.ibm.streamsx.hdfs.HDFS2FileSource.initialize(HDFS2FileSource.java:119)
at
com.ibm.streams.operator.internal.runtime.api.OperatorAdapter.initialize(OperatorAdapte
r.java:735)
at com.ibm.streams.operator.internal.jni.JNIBridge.<init>(JNIBridge.java:271)
Caused by: javax.security.auth.login.FailedLoginException: Null key
at
com.ibm.security.jgss.i18n.I18NException.throwFailedLoginException(I18NException.java:3
2)
at com.ibm.security.auth.module.Krb5LoginModule.a(Krb5LoginModule.java:722)
at com.ibm.security.auth.module.Krb5LoginModule.b(Krb5LoginModule.java:154)
at com.ibm.security.auth.module.Krb5LoginModule.login(Krb5LoginModule.java:411)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:95)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:55)
```

To resolve this problem, ensure the following:

1. If IBM Streams is not installed on one of the Hadoop cluster nodes, copy the `/usr/iop/4.2.0.0/hadoop` and `/usr/iop/4.2.0.0/hadoop-hdfs` directories from one of the cluster nodes to the `/home/streamsadmin/Hadoop/` directory on the IBM Streams server.
2. Edit the streamsadmin user `.bashrc` file to include the following line: `export HADOOP_HOME=/home/streamsadmin/Hadoop/hadoop`
3. Copy the `/etc/krb5.conf` file from the KDC computer to the computer where IBM Streams is installed.
4. Install [IBM Streams Fix Pack 4.2.0.3](http://www.ibm.com/support/docview.wss?uid=swg21997273) (www.ibm.com/support/docview.wss?uid=swg21997273).
 - Check the permissions for `ibmjgssprovider.jar`.
 - Make a backup copy of `ibmjgssprovider.jar` in a different location than the original. Ensure that you place the backup copy outside of the JVM path and outside of directories in the classpath.
 - Delete the original `ibmjgssprovider.jar` file. Do not rename it. You must delete the file.
 - Copy the backup file to the location of the original file. Ensure that the permissions for the file are the same as were set for the original `ibmjgssprovider.jar` file.
 - Restart the application.

Caused by: java.lang.ClassNotFoundException: com.ibm.security.pkcs7.Content

You are running the `./processvoice.sh voicemetadata.csv` command, and you receive the following error:

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
10000, (+1)-204-353-7282, dev004, 10002; 10004, (+1)-687-225-8261;
(+1)-395-309-9915, dev002; dev003, Audio1.wav,
2017-04-14, 22:08:30, 22:09:19, POL1, gcid001571
Exception in thread "main" java.lang.NoClassDefFoundError: com/ibm/security/pkcs7/
Content
at VoiceKafkaClient.main(VoiceKafkaClient.java:67)
Caused by: java.lang.ClassNotFoundException: com.ibm.security.pkcs7.Content
at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:331)
at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
... 1 more
```

To resolve this problem, ensure that you include the `ibmpkcs.jar` file in the classpath.

No Audio files to play in UI – Voice

If you receive this error, ensure that the relevant audio file is available in the `installedApps/SurveillanceWorkbench.ear/SurveillanceWorkbench.war` directory where WebSphere Application Server is installed. The audio file should be named `commid.wav`, which is determined by the `sifs.communication` table.

org.apache.kafka.common.KafkaException: Failed to construct kafka producer

While you are running the Process Alert Streams job, you receive the following error:

```
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Exception in thread "Thread-11" org.apache.kafka.common.KafkaException: Failed to
construct kafka producer
at org.apache.kafka.clients.producer.KafkaProducer.<init>(KafkaProducer.java:335)
at org.apache.kafka.clients.producer.KafkaProducer.<init>(KafkaProducer.java:188)
at
com.ibm.streamsx.messaging.kafka.ProducerByteHelper.<init>(KafkaProducerClient.java:99)
at
com.ibm.streamsx.messaging.kafka.KafkaProducerFactory.getClient(KafkaProducerFactory.ja
va:24)
at com.ibm.streamsx.messaging.kafka.KafkaSink.getNewProducerClient(KafkaSink.java:106)
at com.ibm.streamsx.messaging.kafka.KafkaSink.initialize(KafkaSink.java:97)
at
com.ibm.streamsx.operator.internal.runtime.api.OperatorAdapter.initialize(OperatorAdapte
r.java:735)
at com.ibm.streamsx.operator.internal.jni.JNIBridge.<init>(JNIBridge.java:271)
Caused by: org.apache.kafka.common.KafkaException:
org.apache.kafka.common.KafkaException: java.io.FileNotFoundException:
/home/SIUser/SIKafkaServerSSLKeystore.jks (No such file or directory)
at
org.apache.kafka.common.network.SslChannelBuilder.configure(SslChannelBuilder.java:44)
at org.apache.kafka.common.network.ChannelBuilders.create(ChannelBuilders.java:70)
at org.apache.kafka.clients.ClientUtils.createChannelBuilder(ClientUtils.java:83)
at org.apache.kafka.clients.producer.KafkaProducer.<init>(KafkaProducer.java:277)
... 7 more
Caused by: org.apache.kafka.common.KafkaException: java.io.FileNotFoundException: /
home/SIUser/SIKafkaServerSSLKeystore.jks
(No such file or directory)
```

```

at org.apache.kafka.common.security.ssl.SslFactory.configure(SslFactory.java:110)
at
org.apache.kafka.common.network.SslChannelBuilder.configure(SslChannelBuilder.java:41)
... 10 more
Caused by: java.io.FileNotFoundException: /home/SIUser/SIKafkaServerSSLKeystore.jks
(No such file or directory)
at java.io.FileInputStream.open(FileInputStream.java:212)
at java.io.FileInputStream.<init>(FileInputStream.java:152)
at java.io.FileInputStream.<init>(FileInputStream.java:104)
at org.apache.kafka.common.security.ssl.SslFactory
$SecurityStore.load(SslFactory.java:205)
at org.apache.kafka.common.security.ssl.SslFactory$SecurityStore.access
$000(SslFactory.java:190)
at
org.apache.kafka.common.security.ssl.SslFactory.createSSLContext(SslFactory.java:126)
at org.apache.kafka.common.security.ssl.SslFactory.configure(SslFactory.java:108)
... 11 more

```

To resolve this problem, ensure that the jks file is correct in the /home/streamsadmin/config/producer.propertiesfile.

CDISR5030E: An exception occurred during the execution of the PerfLoggerSink operator

While you are running the pump-and-dump job, you receive the following error:

```

15 May 2017 12:45:54.464 [7178] ERROR
#splapptrc,J[14],P[20],PerfLoggerSink,spl_function
M[PerfLoggerSink.cpp:Helper:125] - CDISR5023E: The /home/streamsadmin/PNDDData/PnDPerf-
trades.log output file
did not open. The error is: No such file or directory.
15 May 2017 12:45:54.542 [7178] ERROR
#splapptrc,J[14],P[20],PerfLoggerSink,spl_operator
M[PEImpl.cpp:instantiateOperators:664] - CDISR5030E: An exception occurred during the
execution of the
PerfLoggerSink operator. The exception is: The /home/streamsadmin/PNDDData/PnDPerf-
trades.log output file
did not open. The error is: No such file or directory.
15 May 2017 12:45:54.542 [7178] ERROR #splapptrc,J[14],P[20],PerfLoggerSink,spl_pe
M[PEImpl.cpp:process:1319]
- CDISR5079E: An exception occurred during the processing of the processing element.
The error is:
The /home/streamsadmin/PNDDData/PnDPerf-trades.log output file did not open. The error
is: No such file or directory..
15 May 2017 12:45:54.543 [7178] ERROR
#splapptrc,J[14],P[20],PerfLoggerSink,spl_operator M[PEImpl.cpp:process:1350]
- CDISR5053E: Runtime failures occurred in the following operators: PerfLoggerSink

```

To resolve this problem, ensure that you create the folder that is set for the DATAPATH value in the start_pnd.sh file before you run the Streams job.

For example, DATAPATH=/home/streamsadmin/PNDDData.

CDISR5023E: The error is: No such file or directory

While you are running the Spoofing Streams job, you receive the following error:

```

15 May 2017 12:48:02.055 [15067] ERROR #splapptrc,J[22],P[28],FileSink_7,spl_function
M[FileSink_7.cpp:Helper:125] - CDISR5023E: The /home/streamsadmin/testfolders/
spoofingLog.txt output file
did not open. The error is: No such file or directory.

```

```

15 May 2017 12:48:02.131 [15067] ERROR #splapptrc,J[22],P[28],FileSink_7,spl_operator
M[PEImpl.cpp:instantiateOperators:664] - CDISR5030E: An exception occurred during the
execution of the
FileSink_7 operator. The exception is: The /home/streamsadmin/testfolders/
spoofingLog.txt output file did
not open. The error is: No such file or directory.
15 May 2017 12:48:02.131 [15067] ERROR #splapptrc,J[22],P[28],FileSink_7,spl_pe
M[PEImpl.cpp:process:1319]
- CDISR5079E: An exception occurred during the processing of the processing element.
The error is:
The /home/streamsadmin/testfolders/spoofingLog.txt output file did not open. The
error is: No such file or directory..

```

To resolve this problem, check if the correct values for the -C data-directory value when you submit the spoofing job, and that the folder exists before you run the job.

For example, -C data-directory=/home/streamsadmin/testfolders.

java.lang.Exception: 404 response code received from REST service!!

While you are running the e-comms Streams job, you receive the following error:

In the Streams error log:

```

Exception in thread "Thread-52" java.lang.Exception: 404 response code received from
REST service!!
at com.ibm.sifs.ecomm.Utility.invokeService(Utility.java:101)
at com.ibm.sifs.ecomm.PolicyLoader.loadAllPolicies(PolicyLoader.java:211)
at com.ibm.sifs.ecomm.PolicyLoader.process(PolicyLoader.java:157)
at
com.ibm.streams.operator.internal.runtime.api.OperatorAdapter.processTuple(OperatorAdap
ter.java:1591)
at com.ibm.streams.operator.internal.runtime.api.OperatorAdapter
$1.tuple(OperatorAdapter.java:825)
at com.ibm.streams.operator.internal.runtime.api.OperatorAdapter
$1.tuple(OperatorAdapter.java:818)
at com.ibm.streams.operator.internal.ports.SwitchingHandler
$Alternate.tuple(SwitchingHandler.java:162)
at
com.ibm.streams.operator.internal.network.DeserializingStream.tuple(DeserializingStream
.java:76)
at
com.ibm.streams.operator.internal.network.DeserializingStream.processRawTuple(Deseriali
zingStream.java:65)
at
com.ibm.streams.operator.internal.runtime.api.InputPortsConduit.processRawTuple(InputPo
rtsConduit.java:100)
at com.ibm.streams.operator.internal.jni.JNIBridge.process(JNIBridge.java:312)
Couldn't open cc.mallet.util.MalletLogger resources/logging.properties file.

```

In the WebSphere Application Server logs:

```

[5/15/17 14:23:43:298 IST] 000003c3 WebContainer E
com.ibm.ws.webcontainer.internal.WebContainer handleRequest
SRVE0255E: A WebGroup/Virtual Host to handle /CommServices/ecomms/policy/ has not been
defined.

```

Ensure that the com_ibm_sifs_service_data-2_0_0-SNAPSHOT_war file is deployed to WebSphere Application Server in detailed mode, and that the user role is mapped correctly as an admin user.

java.lang.NoClassDefFoundError: com/ibm/si/security/kafka/SISecureKafkaProducer

The following error messages appear in the WebSphere Application Server log files:

```
[5/15/17 15:04:19:526 IST] 000000e4 SIKafkaPolicy I {security.protocol=SSL,
serializer.class=kafka.serializer.StringEncoder,
ssl.keystore.password=SurveillanceInsightPass2016, ssl.truststore.location=/home/
sifsuser/SIKafkaServerSSLTruststore.jks, ssl.truststore.type=JKS,
ssl.enabled.protocols=TLSv1.1, bootstrap.servers=9.194.241.131:9093,
ssl.truststore.password=SurveillanceInsightPass2016,
value.serializer=org.apache.kafka.common.serialization.ByteArraySerializer,
request.required.acks=1,
ssl.keystore.location=/home/sifsuser/SIKafkaServerSSLKeystore.jks,
key.serializer=org.apache.kafka.common.serialization.StringSerializer,
ssl.protocol=TLSv1.1,
ssl.key.password=SIKafkaKeyPass}==={com.ibm.si.encryption.key.length=128,
com.ibm.si.encryption.keystore.location=/home/sifsuser/SIKafkaEncrypt.jks,
com.ibm.si.encryption.keystore.password=SurveillanceInsightPass2016,
com.ibm.si.encryption.certificate.alias=SIKafkaSecurityKey,
com.ibm.si.encryption.algorithm.name=AES}
[5/15/17 15:04:19:526 IST] 000000e4 ServletWrappe E
com.ibm.ws.webcontainer.servlet.ServletWrapper init
Uncaught.init.exception.thrown.by.servlet
[5/15/17 15:04:19:527 IST] 000000e4 webapp E com.ibm.ws.webcontainer.webapp.WebApp
logServletError SRVE0293E: [Servlet Error]-[SIDataRESTApp]:
java.lang.NoClassDefFoundError: com/ibm/si/security/kafka/SISecureKafkaProducer
at
com.ibm.sifs.service.data.policy.SIKafkaPolicyProducer.initializeProducer(SIKafkaPolicy
Producer.java:67)
at com.ibm.sifs.service.SIDataRESTServlet.init(SIDataRESTServlet.java:44)
at javax.servlet.GenericServlet.init(GenericServlet.java:244)
at com.ibm.ws.webcontainer.servlet.ServletWrapper.init(ServletWrapper.java:341)
at com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.init(ServletWrapperImpl.java:168)
at
com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:633)
at
com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:477)
at
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest(ServletWrapperImpl.jav
a:178)
at
com.ibm.ws.webcontainer.filter.WebAppFilterManager.invokeFilters(WebAppFilterManager.ja
va:1124)
at
com.ibm.ws.webcontainer.servlet.CacheServletWrapper.handleRequest(CacheServletWrapper.j
ava:82)
at com.ibm.ws.webcontainer.WebContainer.handleRequest(WebContainer.java:961)
at com.ibm.ws.webcontainer.WSWebContainer.handleRequest(WSWebContainer.java:1817)
at com.ibm.ws.webcontainer.channel.WCChannelLink.ready(WCChannelLink.java:294)
at
com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleDiscrimination(HttpInboundLi
nk.java:465)
at
com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleNewRequest(HttpInboundLink.j
ava:532)
at
com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.processRequest(HttpInboundLink.jav
a:318)
at
com.ibm.ws.http.channel.inbound.impl.HttpICLReadCallback.complete(HttpICLReadCallback.j
ava:88)
at com.ibm.ws.ssl.channel.impl.SSLReadServiceContext
$SSLReadCompletedCallback.complete(SSLReadServiceContext.java:1820)
```

```

at
com.ibm.ws.tcp.channel.impl.AioReadCompletionListener.futureCompleted(AioReadCompletion
Listener.java:175)
at com.ibm.io.async.AbstractAsyncFuture.invokeCallback(AbstractAsyncFuture.java:217)
at
com.ibm.io.async.AsyncChannelFuture.fireCompletionActions(AsyncChannelFuture.java:161)
at com.ibm.io.async.AsyncFuture.completed(AsyncFuture.java:138)
at com.ibm.io.async.ResultHandler.complete(ResultHandler.java:204)
at com.ibm.io.async.ResultHandler.runEventProcessingLoop(ResultHandler.java:775)
at com.ibm.io.async.ResultHandler$2.run(ResultHandler.java:905)
at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1892)

```

To resolve this problem, check for the following in the `producer.properties` file:

- The `/home/sifsuser/producer.properties` has the correct jks files, and the jks files exist in that location.
- The WebSphere Application Server JNDI variable for the `KAFKA_ENCRYPTION_PROP_LOC` property is set and the `kafka_encryption.properties` file has the correct `algorithm.name` and `key.length` properties set.

For example,

```

com.ibm.si.encryption.algorithm.name=3DES
com.ibm.si.encryption.key.length=128
com.ibm.si.encryption.keystore.location=/home/sifsuser/SIKafkaEncrypt.jks
com.ibm.si.encryption.keystore.password=SurveillanceInsightPass2016
com.ibm.si.encryption.certificate.alias=SIKafkaSecurityKey

```

- The `com.ibm.sifs.security-2.0.0-SNAPSHOT.jar` file is in the `/opt/IBM/WebSphere/AppServer/lib/ext` directory.
- The `SIFServices.war` is deployed to WebSphere Application Server in detailed mode and the Solr shared library jar is included in the deployment.

java.lang.NumberFormatException: For input string: "e16"

While you are running the Spoofing Spark job, you receive the following error:

```

17/05/17 15:28:53 INFO yarn.ApplicationMaster: Prepared Local resources
Map(__spark_conf__
-> resource { scheme: "hdfs" host: "cimkc2b094.in.ibm.com" port: 8020 file:
"/user/sifsuser/.sparkStaging/application_1494933278396_0014/__spark_conf__.zip" }
size: 93300 timestamp:
1495015118409 type: ARCHIVE visibility: PRIVATE, __spark_libs__/spark-
yarn_2.11-2.0.1.jar -> resource {
scheme: "hdfs" host: "cimkc2b094.in.ibm.com" port: 8020 file:
"/user/sifsuser/.sparkStaging/application_1494933278396_0014/spark-
yarn_2.11-2.0.1.jar" } size: 669595
timestamp: 1495015118328 type: FILE visibility: PRIVATE)
17/05/17 15:28:53 ERROR yarn.ApplicationMaster: Uncaught exception:
java.lang.IllegalArgumentException: Invalid ContainerId:
container_e16_1494933278396_0014_01_000001
at org.apache.hadoop.yarn.util.ConverterUtils.toContainerId(ConverterUtils.java:182)
at
org.apache.spark.deploy.yarn.YarnSparkHadoopUtil.getContainerId(YarnSparkHadoopUtil.sca
la:187)
at org.apache.spark.deploy.yarn.YarnRMClient.getAttemptId(YarnRMClient.scala:95)
at org.apache.spark.deploy.yarn.ApplicationMaster.run(ApplicationMaster.scala:184)
at org.apache.spark.deploy.yarn.ApplicationMaster$$anonfun$main$1.apply$mcV
$sp(ApplicationMaster.scala:749)
at org.apache.spark.deploy.SparkHadoopUtil$$anon$1.run(SparkHadoopUtil.scala:71)
at org.apache.spark.deploy.SparkHadoopUtil$$anon$1.run(SparkHadoopUtil.scala:70)
at java.security.AccessController.doPrivileged(Native Method)
at javax.security.auth.Subject.doAs(Subject.java:422)
at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1491)
at org.apache.spark.deploy.SparkHadoopUtil.runAsSparkUser(SparkHadoopUtil.scala:70)

```

```

at org.apache.spark.deploy.yarn.ApplicationMaster$.main(ApplicationMaster.scala:747)
at org.apache.spark.deploy.yarn.ApplicationMaster.main(ApplicationMaster.scala)
Caused by: java.lang.NumberFormatException: For input string: "e16"
at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
at java.lang.Long.parseLong(Long.java:589)
at java.lang.Long.parseLong(Long.java:631)
at
org.apache.hadoop.yarn.util.ConverterUtils.toApplicationAttemptId(ConverterUtils.java:137)
at org.apache.hadoop.yarn.util.ConverterUtils.toContainerId(ConverterUtils.java:177)
... 12 more
17/05/17 15:28:53 INFO yarn.ApplicationMaster: Final app status: FAILED, exitCode: 10,
(reason:
Uncaught exception: java.lang.IllegalArgumentException: Invalid ContainerId:
container_e16_1494933278396_0014_01_000001)
17/05/17 15:28:53 INFO util.ShutdownHookManager: Shutdown hook called

```

To resolve this problem, ensure that the JAR file names and paths are correct when you run the Spark submit command.

CDISP0180E ERROR: The error is:The HADOOP_HOME environment variable is not set

While you are running the Spoofing Streams job, you receive the following error:

```

/opt/ibm/InfoSphere_Streams/4.2.0.2/toolkits/com.ibm.streamsx.hdfs/com.ibm.
streamsx.hdfs/HDFS2DirectoryScan/HDFS2DirectoryScan: CDISP0180E ERROR: An error
occurred
while the operator model was loading. The error is: The HADOOP_HOME environment
variable is not set.

```

To resolve this problem, ensure that the HADOOP_HOME variable is set to the Hadoop installation directory in the terminal where the Streams jobs are run.

For example, export HADOOP_HOME=/usr/iop/4.2.0.0/hadoop.

Database connectivity issues

If you receive database connectivity errors, try the following:

Change the DB2 port from 50000 to 50001, and then perform the following steps:

- Use the following command to determine on which port IBM DB2 is listening: `grep DB2_db2inst1 /etc/services`.
- Modify the port number in `/etc/services` to point to 50001.
For example, `db2c_db2inst1 50001/tcp`
- Run the `db2set -all` command. If the command shows TCP for the DB2COMM variable, change the value to SSL.
- Restart the database and run `netstat` and `db2set -all`.

status:"500","message":"Expecting '{' on line 1, column 0 instead, obtained token: 'Token:'

While you are running the e-comm Curl commands, you receive the following error:

```

[5/16/17 10:11:48:483 IST] 000000e9 SIPolicyResou E Datasource is null
com.ibm.sifs.service.util.SIServiceException: Datasource is null
at com.ibm.sifs.service.util.SIDBUtil.getConnection(SIDBUtil.java:79)
at
com.ibm.sifs.service.data.policy.SIPolicyResource.getAllPolicies(SIPolicyResource.java:
250)

```

```

at sun.reflect.GeneratedMethodAccessor59.invoke(Unknown Source)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:55)
at java.lang.reflect.Method.invoke(Method.java:508)
at
org.apache.wink.server.internal.handlers.InvokeMethodHandler.handleRequest(InvokeMethod
Handler.java:63)
at
org.apache.wink.server.handlers.AbstractHandler.handleRequest(AbstractHandler.java:33)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
6)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
2)
at
org.apache.wink.server.handlers.AbstractHandlersChain.doChain(AbstractHandlersChain.jav
a:75)
at
org.apache.wink.server.internal.handlers.CreateInvocationParametersHandler.handleReques
t
(CreateInvocationParametersHandler.java:54)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
6)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
2)
at
org.apache.wink.server.handlers.AbstractHandlersChain.doChain(AbstractHandlersChain.jav
a:75)
at
org.apache.wink.server.handlers.AbstractHandler.handleRequest(AbstractHandler.java:34)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
6)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
2)
at
org.apache.wink.server.handlers.AbstractHandlersChain.doChain(AbstractHandlersChain.jav
a:75)
at
org.apache.wink.server.internal.handlers.FindResourceMethodHandler.handleResourceMethod
(FindResourceMethodHandler.java:151)
at org.apache.wink.server.internal.handlers.FindResourceMethodHandler.handleRequest
(FindResourceMethodHandler.java:65)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
6)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
2)
at
org.apache.wink.server.handlers.AbstractHandlersChain.doChain(AbstractHandlersChain.jav
a:75)
at org.apache.wink.server.internal.handlers.FindRootResourceHandler.handleRequest
(FindRootResourceHandler.java:95)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
6)
at
org.apache.wink.server.handlers.RequestHandlersChain.handle(RequestHandlersChain.java:2
2)
at
org.apache.wink.server.handlers.AbstractHandlersChain.doChain(AbstractHandlersChain.jav

```

```
a:75)
{"status":"500","message":"Expecting '{' on line 1, column 0 instead, obtained token:
'Token: EOF'"}
}
```

To resolve this problem, ensure that the path of the `policy.json` file is correct in the curl command.

For example,

```
curl -k -H 'Content-Type: application/json' -H 'source:Actiance' -X POST
--data-binary @/media/Installable/base/Database/Sample_Data/EComm/Policy/policy.json -
v
--user ibmrest1:ibmrest@pwd1 --digest https://server_name:9443/CommServices/ecomm/
policy
```

Runtime failures occurred in the following operators: FileSink_7

While you are running the Spoofing Streams job, you receive the following error:

```
ERROR ::PEC.StartPE M[PECServer.cpp:runPE:282] P[25] - PE 25 caught Distillery
Exception: 'virtual void SPL::PEImpl::process()' [./src/SPL/Runtime/ProcessingEleme
nt/PEImpl.cpp:1351] with msg: Runtime failures occurred in the following operators:
FileSink_7.
```

To resolve this problem, ensure that the following folders exist on the Hadoop file system before you run the spoofing job.

```
drwxr-xr-x - sifsuser hadoop 0 date time /user/sifsuser/execution
drwxr-xr-x - sifsuser hadoop 0 date time /user/sifsuser/order
drwxr-xr-x - sifsuser hadoop 0 date time /user/sifsuser/quote
drwxr-xr-x - sifsuser hadoop 0 date time /user/sifsuser/trade
```

Subject and email content is empty

To resolve this problem, ensure that the `commevidence` directory is copied to the doc root of IBM HTTP Server.

Appendix A. Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products.

For information about the commitment that IBM has to accessibility, see the [IBM Accessibility Center](http://www.ibm.com/able) (www.ibm.com/able).

HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

Notices

This information was developed for products and services offered worldwide.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing
3755 Riverside Dr.

Ottawa, ON
K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

Trademarks

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at www.ibm.com/legal/copytrade.shtml.

Index

A

accessibility [123](#)
architecture [1](#)
audio metadata schema [25](#)

B

bulk execution detection [25](#)
bulk order detection [25](#)

C

chat data [35](#)
classifier library [66](#)
cognitive analysis and reasoning component [1](#)
compliance workbench [1](#)
concept mapper [61](#)
concept mapper library [63](#)

D

data ingestion
 e-comm data [38](#)
data store [1](#)
document classifier [61](#)

E

e-comm data ingestion [38](#)
e-comm surveillance [35](#)
email data [35](#)
emotion detection [61](#)
emotion detection library [61](#)
end of day schema [23](#)
event data schema [24](#)
event schema [24](#)
execution schema [18](#)

G

guidelines for new models [31](#)

H

high order cancellation [25](#)

I

indexing [75](#)
inference engine
 risk model [71](#)
 running the inference engine [73](#)
introduction [v](#)

M

models
 guidelines [31](#)

N

natural language libraries
 classifier [66](#)
 concept mapper [63](#)
 emotion detection [61](#)

O

order schema [20](#)
overview [1](#)

P

price trend [25](#)
pump and dump
 use case [28](#)

Q

quote schema [21](#)

R

real-time analytics [1](#)
risk model
 inference engine [71](#)
running the inference engine [73](#)

S

schemas
 audio metadata [25](#)
 end of day [23](#)
 event [24](#)
 event data [24](#)
 execution [18](#)
 order [20](#)
 quote [21](#)
 ticker price [18](#)
 trade [23](#)
searching [75](#)
solution architecture [1](#)
spoofing
 use case [30](#)

T

ticker price schema [18](#)
trade schema [23](#)
trade surveillance component [17](#)

trade surveillance toolkit [25](#)

U

use case

 pump and dump [28](#)

 spoofing [30](#)

V

voice surveillance [55](#)

