

IBM Surveillance Insight for Financial Services
Version 1.2.0

*IBM Surveillance Insight for Financial
Services Solution Guide*

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 67.

Product Information

This document applies to Version 1.2.0 and may also apply to subsequent releases.

Licensed Materials - Property of IBM

© **Copyright IBM Corporation 2016.**

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
Components of the solution	4
The IBM Surveillance Insight for Financial Services solution architecture	5
Chapter 2. Load data	9
Loading structured data	9
Setting up the sample unstructured data	9
E-Comm data ingestion	11
Email data format	15
Voice data format	17
Chapter 3. Use the product	19
The Login page	19
The home page	19
Trade analytics	20
The Pump and Dump Reasoning page	20
The Spoofing Charts page	23
Trade Analysis page	23
Live Monitoring page	25
Know Your Trades page	26
Know Your Employee page	26
E-communication analytics	26
Running a pump and dump analysis	28
Chapter 4. Trade Surveillance Toolkit	29
Pump and Dump use case	32
Spoofing detection use case	34
Guidelines for developing new models	34
Chapter 5. NLP libraries	37
Emotion Detection library	37
Concept Mapper library	39
Classifier library	42
Chapter 6. Policy	47
Alert services and policy execution	48
Chapter 7. E-comm schema and persistence overview	51
Chapter 8. Data schemas	53
Ticker price schema	53
Execution schema	53
Order schema	55
Quote schema	56
Trade schema	58
End of day (EOD) schema	59
Event schema	60
Event data schema	60
Audio metadata schema	60
Policy schema	61
Communication object tuple	62

Chapter 9. Voice surveillance	63
Appendix. Accessibility features.	65
Notices	67
Index	71

Introduction

The IBM® Surveillance Insight for Financial Services solution uses cognitive and advanced analytic capabilities for real-time supervision and surveillance, enforcement of regulatory rules, and proactive monitoring of fraudulent activities.

IBM Surveillance Insight for Financial Services uses IBM advanced analytics, including graph computing, big data analytics and reasoning, natural language processing, sentiment analysis, stream analysis, and semantic event analysis. It includes a large-scale real-time graphing tool that can handle and analyze trillions of linked pieces of data. A novel reasoning engine and analytics pipeline to enable it to handle various cognitive tasks. Natural language processing capability enables it to understand text information and to distinguish ambiguous terms based on context and knowledge graphs. It includes sentiment analysis capabilities to understand emotions. Stream analysis provides filters to handle very large amounts of streaming data such as trades. Semantic event analysis applies machine learning to map anomaly signals to human events.

You use this solution to

- Identify new trading and behavioral patterns
- Combine information from multiple data sources, such as trades and electronic communications
- Apply learning models, which are continuously updated
- Provide effective and easy to use visualization tools

The solution provides the following outcomes:

- Identification of new predictive patterns by linking various information with Trader profiles, such as "know your trader"
- Improving the accuracy of alerts by using risk profiling, such as reducing false positives and false negatives
- Improving the efficiency of investigations

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 (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 "Accessibility features," on page 65.

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.

IBM Surveillance Insight for Financial Services provides trade and electronic communications (eComms) surveillance components. On the trade side, the solution generates alerts from structured trade data (such as order data, trade data, and executions data) by detecting patterns through a high-speed and scalable transaction analysis system that can process large volumes of trade data. Factors that contribute to anomalous behavior can include trading behavior anomalies as compared to peers and an individual's past trading history, communication anomalies such as discussing something suspicious before or after trades, and outside public events from news and social media. However, looking at the trade side alone is not enough to conclude that there is evidence of wrong-doing.

By including eComms surveillance the solution can detect the anomalies and correlate them with the trading activity. Combining trade and eComms surveillance presents a holistic picture. Additionally, the solution provides reason based on data observations that are made over a long time period. As well, the solution learns anomalies from data, by comparing traders' behavior with self, population, and affinity groups. This cognitive reasoning engine also hierarchically correlates the causality of semantic events to predict potential risks.

For the eComms surveillance, the solution can understand natural language and detect various signals from it. For instance, the solution can determine the sentiment and the semantics of eComms communication.

A key piece of the solution is to make it intuitive for compliance teams to easily view the scores of evidence data that is gathered and allow them to explore the data to ensure that there is evidence to pursue next steps. IBM Surveillance Insight for Financial Services provides different visualization techniques that can handle large, multivariate, and heterogeneous graphs and transition network graphs to provide the most easily understandable representation of the underlying evidence data.

With the trade and eComms surveillance components and the cognitive capabilities, IBM Surveillance Insight for Financial Services can meet the supervision and surveillance needs of the financial services industry across front, mid, and back-office functions.

Solution capabilities and features

Taking the surveillance to the next level, IBM Surveillance Insight for Financial Services offers the first of a kind—a cognitive reasoning engine which goes beyond a rule-based alert system and leverages real-time trader trading behaviors along with their communications and social network through email, social media, blogs,

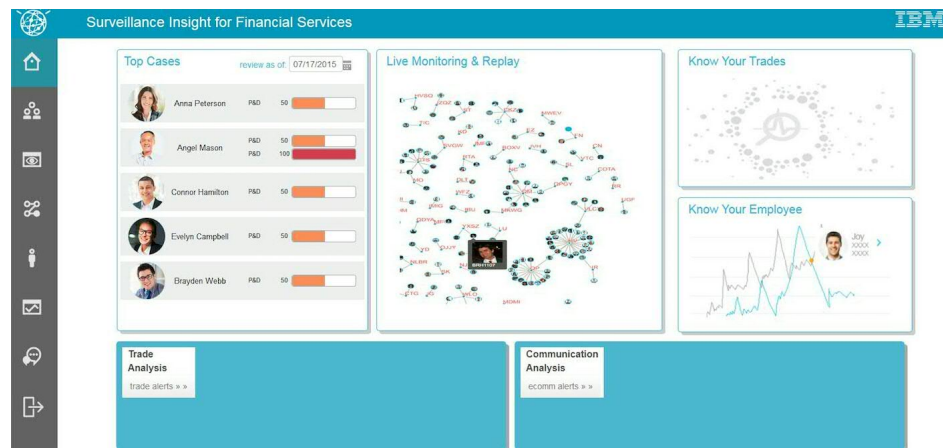
and tweets to provide more accurate and actionable insights and alerts. The solution offers real-time surveillance capability through its reasoning engine with unique visualization experience.

The IBM Surveillance Insight for Financial Services solution provides online, real-time business activity monitoring, with powerful capabilities for helping financial market firms use advanced data mining, statistical models, and sophisticated pattern mining to identify and act on potentially fraudulent activity. In addition, these patterns can be used to generate alerts. The solution takes advantage of IBM's substantial research and development investments to support a real-time surveillance infrastructure that enables firms to:

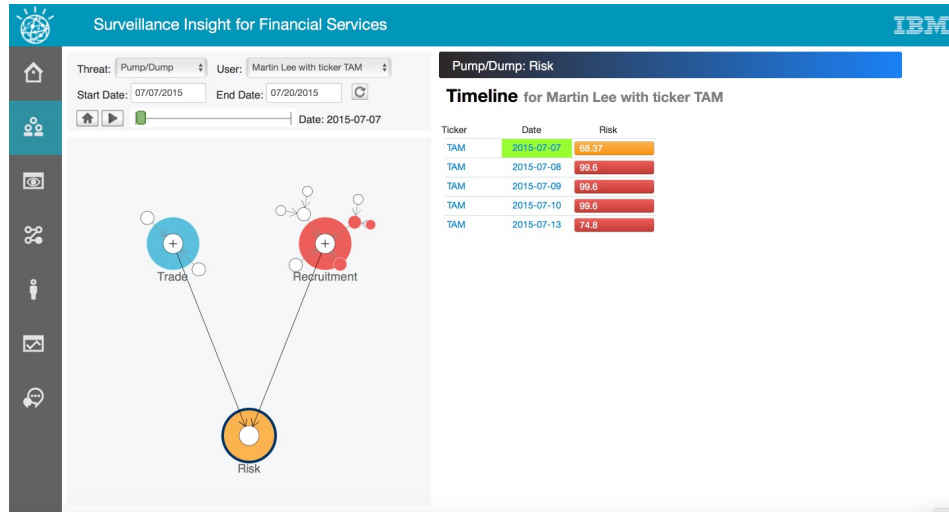
- Use advanced pattern mining capabilities and statistical models to review market information and identify exceptions within large volumes of data. Leverage cognitive networks that are fundamental form of brains—large-scale Markovian and Bayesian network tools and deep learning tools—to identify surveillance insights.
- Use industry's first ever reasoning engine to combine traditional structured data analysis with unstructured data analysis (such as semantic analysis, behavioral analysis, and emotional analysis) to provide evidence of suspicious transactions.
- Use advanced graph visualization and reasoning that is combined with traditional drill-down and drill-across techniques to help with investigation.

Features such as unstructured data analysis, identity resolution, event correlation, cognitive reasoning, and knowledge management help firms identify scenarios and exceptions for immediate qualification, investigation, and correction. Advanced data mining capabilities help identify suspicious trading behaviors.

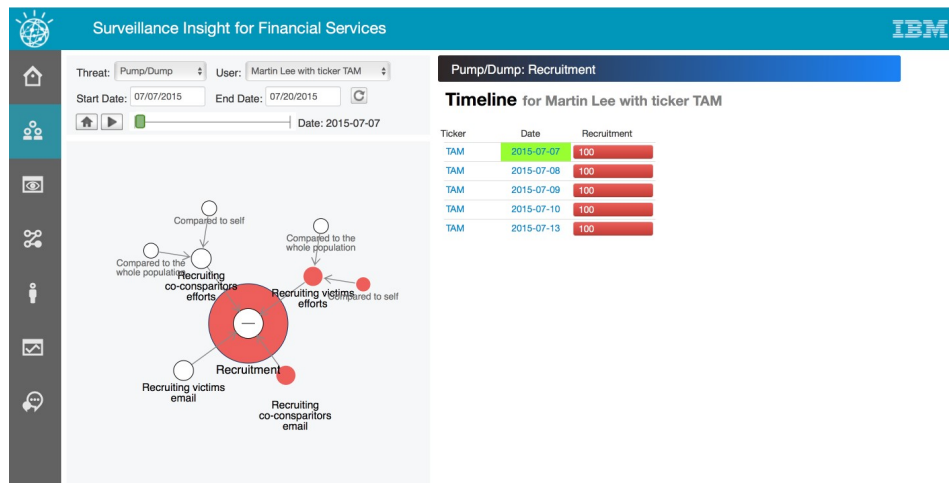
The solution interface



The solution uses integrated analytics to combine trade data with electronic communications to help improve efficiency and accuracy of surveillance results. It applies a reasoning engine to understand traders' long-term behavior, detect anomalies from various 'weak signals' and provide risk estimation to compliance officers.



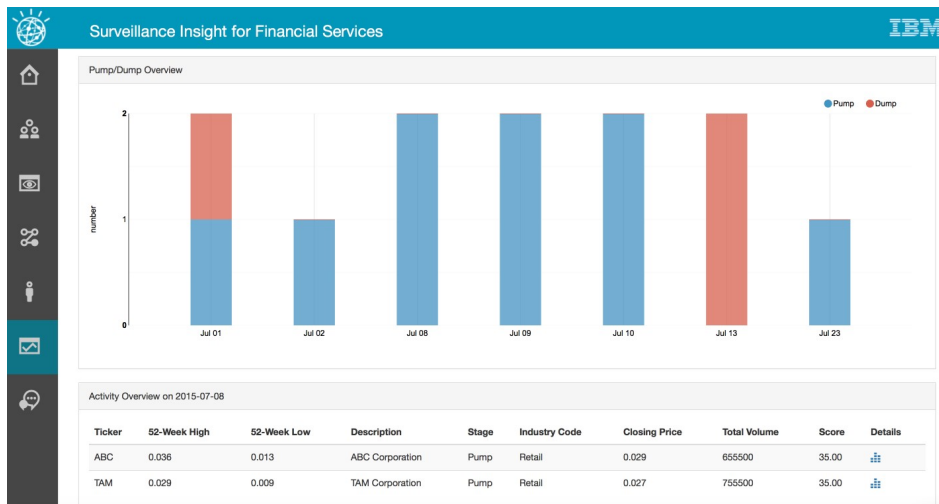
IBM Surveillance Insight for Financial Services learns anomalies from data, by comparing traders' behavior with self, population, and affinity groups. This cognitive reasoning engine also hierarchically correlates the causality of semantic events to predict the potential upcoming risk. Information and alerts are delivered through effective and easy-to-use visualization tools.



The solution uses large-scale graph computing technology to handle and analyze large and linked data sets to gain insights into trading behavior.



Analysis is provided in real time and can scale to handle large volumes of transactions.



Components of the solution

IBM Surveillance Insight for Financial Services offers mid and back office surveillance on market activities and communication in order to detect and report possible market abuse activity. It contains components that include: IBM Base Surveillance Analytics, IBM Electronic Communication Surveillance Analytics, IBM Trade Surveillance Analytics, and IBM Voice Surveillance Analytics.

You must install the IBM Base Surveillance Analytics component before installing other components. To leverage the full capability of the solution it is best to install all components.

IBM Base Surveillance Analytics

This mandatory component offers the Surveillance Platform, which includes IBM Spark, Hadoop, Graph Store, Reasoning Engine, Text analytics framework, Search and Indexing capabilities, and other components that are common for the surveillance framework.

IBM Electronic Communication Surveillance Analytics

This optional component monitors e-communication channels, such as email and instant messaging. It generates alerts based on the policy that you set.

IBM Trade Surveillance Analytics

This optional component monitors the activity of traders to detect market abuse activities. Currently the solution has pre-built models to detect pump-and-dump and spoofing scenarios. In addition, the component offers re-usable operators such as a bulk order detector to detect base alerts.

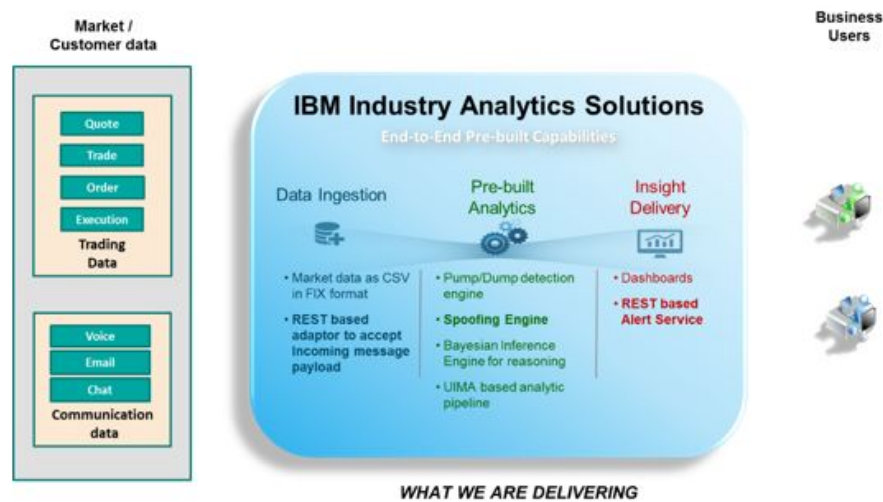
IBM Voice Surveillance Analytics

This optional component monitors the voice communications of traders and employees. It converts the voice to text by using the IBM Watson Speech to Text component and performs analytics on the text to detect and report any possible market abuse patterns. It generates alerts based on the policy that you set.

The IBM Surveillance Insight for Financial Services solution architecture

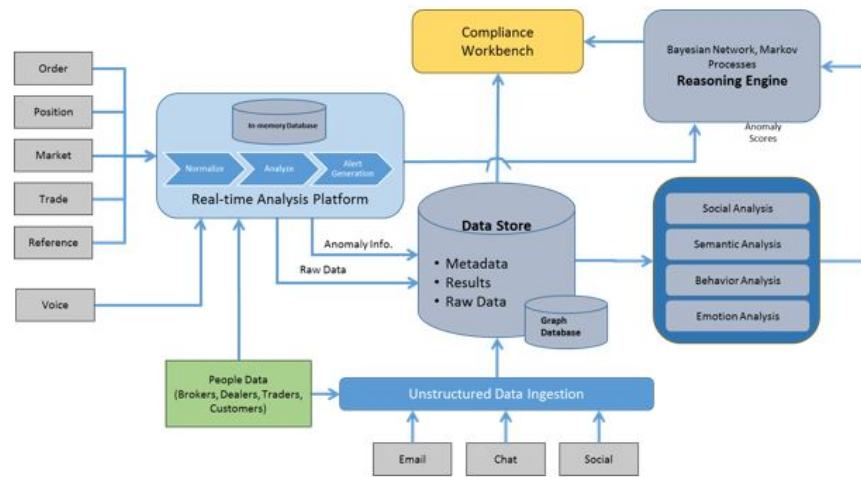
The solution accepts structured market data and unstructured communication data, such as voice, email, and chat messages. It performs real-time and offline analytics on the data and generates relevant alerts for business users. Analytic results are delivered through a web-based user interface.

The following graphic provides a high-level overview of the solution.



The solution architecture provides the following main components:

- Data Ingestion
- Real-Time Analytics
- Cognitive Analysis and Reasoning
- Data Store
- Compliance Workbench



The solution defines the following data ingestion points:

- For trade data, the solution uses a secure FTP-based interface to ingest trade, order, quote, execution, end of the day, and initial positions data. The schema is based on FIX (<http://www.fixtradingcommunity.org/>), which makes it easy to import the data.
- For e-communication data, the solution offers secure REST service over DIGEST authentication. You can import your data and create a payload and ingest the data into the solution.
- For voice data, the solution offers secure FTP and also secure Kafka based messaging. You must upload your voice data using a secure FTP method and send the metadata about the voice and its location to the platform so that it can perform analysis and generate alerts.

Your users can be classified depending on their roles in the company. For example, you might set a policy for traders that detects market-related activities and set a different policy for other employees that generates sentiment and emotion-based alerts.

The solution offers secure RESTful services using DIGEST authentication so that you can manage the policies to be applied on the e-communication and voice data.

Real-time analytics platform

The solution uses the IBM InfoSphere Streams platform for all real-time analytics. The platform analyzes incoming data in real time to detect any possible market abuse patterns or patterns that meet the policy criteria for generating alerts.

The solution can integrate with any case management system that is already deployed in your enterprise. Currently, the solution generates a JSON payload for the alert on a given RESTful service. You must provide the RESTful service that is needed to ingest the cases into the customer case management system.

In the trade component, the solution offers the following features:

- Surveillance Base Toolkit
- Pump and Dump use case
- Spoofing use case

The Surveillance base toolkit includes reusable schemas, operators, and an event model that is used by these operators. Events can be intercepted by the use-case modules to detect use-case specific patterns and generate relevant alerts.

The Pump and Dump and Spoofing use cases are detailed in the Trade Surveillance section.

The real-time analytics module persists the insights that are generated from the data and also takes care of persisting raw data.

All raw data for Trade is persisted in Graph using Graph Adaptor services. All the communication data gets persisted to DB2 tables. All the insights are persisted to DB2 tables.

The Cognitive Analysis and Reasoning component

The solution uses an ADAMS – research component for some of the cognitive analysis requirements and the reasoning engine. The solution also provides custom build libraries for detecting emotions, sentiment, concepts, and classifications.

Data store

The solution contains two different data stores. The graph store implementation that is provided by IBM SystemG Graph is used to store raw market data that is uploaded to the system. The relational database SIFS in DB2, is used to store alerts and configuration information.

Note: IBM System G Graph – is a research asset for providing a scalable data store and for maintaining all the relationships, such as person-person, person-trade, and so on.

The solution uses IBM Solr for indexing the e-communication and voice data. The solution does not persist or maintain any copy of the actual communication. It only indexes the extracted insights that can later be leveraged by the compliance workbench to fetch the necessary evidences for the case. It offers secure REST service for data ingestion as well as for queries.

Compliance Workbench

The solution is intended for use by Surveillance officers or Case officers. It offers a prebuilt compliance workbench to provide a unified view of all the alerts on different traders and employees and also helps the officers to navigate through the case to find details on the market activities as well as the communication activities. The compliance workbench is supported by RESTful services from the back end.

The compliance workbench is deployed in Apache HTTP server and it uses the backend REST services that are deployed in IBM WebSphere Application Server and IBM Graph Server.

Chapter 2. Load data

You can load both structured and unstructured data. Structured data is content such as trade price and order data. Unstructured data is content from trader emails.

The solution requires customer data, such as names and email addresses for processing the E-Comm and Voice data. This data needs to be provided in the form of .csv files. The sample format of the .csv is included with the Solution Sample data (party.csv and party_contact_point.csv). These files need to be loaded into Surveillance tables before any processing for E-Comm or Voice. You can load these files by using any data client tools or use db2 IMPORT command.

In the IBM Surveillance Insight for Financial Services solution, trade data generally refers to stock market data.

Trade data includes the following data:

- Quote Data
- Order Data
- Execution Data
- Trade Data
- End Of Day (EOD) Data

The data must be made available daily in .csv format. Each file contains one day's data. For information about the schema, see "Trade schema" on page 58.

Loading structured data

You must load price and order data into the IBM Surveillance Insight for Financial Services graph database.

The price data is used for the ticker detail graphs, and the order data is used for the order vertex graphs. The price and order data must be in CSV file format.

For more information about the input data files, see Chapter 8, "Data schemas," on page 53.

Procedure

1. To load the price data, copy the CSV file to the *STREAMS_ADAPTOR/data/price* directory.
The *STREAMS_ADAPTOR* location is defined in IBM InfoSphere® Streams user's .bashrc file.
2. To load the order data, copy the CSV file to the *STREAMS_ADAPTOR/data/order* directory.

Setting up the sample unstructured data

Use the following steps to configure the system to load unstructured data, and to test the configuration by loading the sample data that is provided with IBM Surveillance Insight for Financial Services.

Before you begin

The Analytics Content component must already be set up in your cloud environment before you can load the sample data.

Procedure

1. Go to the *<target_directory>* directory, where *<target_directory>* is the directory you entered when you ran `install.py`.
2. Enter the following command to set your environment variables:
`./setenv.sh`
3. Go to the *<target_directory>/eaves-pipeline-pom-1.1-SNAPSHOT-binary* directory.
4. Enter the following command to create an **EAVES** environment variable:
`export EAVES=$(pwd)`
5. Go to the *<target_directory>/eaves-pipeline-pom-1.1-SNAPSHOT-binary/resources/config/jar* directory.
6. Open `run-finance-sample-email-etl.props` in a text editor.
7. Edit the following values to match your environment:
 - a. Change **FILESTORE_dir** to the `IS_FinancialMkts_SurveillanceInsight_1.2/Analytics/FinancialMkts_SurveillanceInsight_DataContent/EmailData` directory. For example, change the value to `/opt/IBM/IS_FinancialMkts_SurveillanceInsight_1.2/Analytics/FinancialMkts_SurveillanceInsight_DataContent/EmailData`.
 - b. Change **EMAIL_STORENAME** value to `emailtable150.csv`, which is the name of the file in the **FILESTORE_dir** directory.
8. Save and close the file.
9. Go to the *<target_directory>/eaves-pipeline-pom-1.1-SNAPSHOT-binary/scripts* directory.
10. Ensure that your Java variables are set.
 - a. Update your **JAVA_HOME** variable to use the Java™ that you installed for WebSphere® Application Server. For example, enter the following command: `export JAVA_HOME=/opt/IBM/WebSphere/AppServer/java/8.0`
 - b. Add the Java bin directory to your **PATH** environment variable. For example, enter the following command: `export PATH=/opt/IBM/WebSphere/AppServer/java/8.0/bin:$PATH`
11. Set the **UIMA_HOME** environment variable to point to the *<target_directory>/opt/apache-uima/bin* directory.
For example, `export UIMA_HOME=/home/IBM_Adams/opt/apache-uima/bin`.
12. Enter the following command to load the data:
`./run-finance-sample-email-etl.sh`
13. After the script is run, you can verify that the data was loaded by using the IBM Surveillance Insight for Financial Services Application Management console.
 - a. In a web browser, enter the following URL:
`http://servername:port/ui-datamgt-birt`
Where *servername* is that name of the server where you installed the Trade Surveillance component, and *port* is the port number that you are using for the WebSphere Application Server profile. The default port number is 9080.
 - b. Click **All Batch Runs**.

- c. Click the batch name for the job that you ran. Each time that you run `./run-finance-sample-email-etl.sh`, a batch name is created based on the time stamp of when you ran the script.

Note: The **Batch Name** value is used in the following step.

- d. Under **Actions**, click **details**.
- e. Click **View Data** to see the data that was loaded.
14. Go to the `<target_directory>/eaves-pipeline-pom-1.1-SNAPSHOT-binary/resources/config/jar` directory.
15. Open `baseline_trade.properties` in a text editor.
16. Change the `ETL_SAMPLE_BATCHNAME_EMAIL` value to be the same as the **Batch Name** value.
17. Save and close the file.
18. Go to the `FinancialMkts_SurveillanceInsight_DataContent/EmailData` directory, and decompress the `emailtext-sample.tar.gz` file. For example, in a terminal window, enter `tar xvf emailtext-sample.tar.gz`
The file is decompressed to a directory that is named `emaildata` in the current location.
19. From the `EmailData` directory, copy the `emailtext` directory to the document root directory for your web server. For example, copy the `emailtext` directory to the `/opt/rh/httpd24/root/var/www/html` directory.
20. Go to the `<target_directory>/eaves-pipeline-pom-1.1-SNAPSHOT-binary/scripts` directory.
21. Enter the following command to combine the analysis of the structured data with the unstructured (email) data:
This step can take some time to run. For example, on a computer with 32 GB of RAM, this step can take 30 - 60 minutes.
`./run-finance-pipeline.sh`

E-Comm data ingestion

The solution provides an adapter for e-comm data. The adapter accepts and parses e-comm data, such as email and chat data. It extracts the necessary elements and covers the information into a communication object tuple that is then processed by the adapter.

The communication object tuple contains following elements:

Table 1. Communication object tuple

Element name	Data type	Purpose
uniqueHash	character	Unique hash value of source communication Id
trader_id	character	Party id of employee
commID	character	email id or chat id
activity_ts	character	Date time of the communication
emailBody	character	Email of Chat content
globalCommId	character	Source communication id
participants	map	Map of email id and its corresponding party id

Table 1. Communication object tuple (continued)

Element name	Data type	Purpose
direction	character	Direction of email, inbound or outbound
outboundLog	character	Outbound log flag
policyId	character	Policy Code
communicationId	character	Communication Id

The solution processes email and chat data in the form of XML files. The sample .xml file for email data is in the following format:

```
<sn:snapshot xmlns:itm-types="http://
dig.apc.xmlns.commons.platform.actiance.com/itm/interaction/types/1.0"
xmlns:sn="http://dig.apc.xmlns.commons.platform.actiance.com/itm/
interaction/snapshot/1.0" xmlns:itm="http://
dig.apc.xmlns.commons.platform.actiance.com/itm/interaction/transcript/
1.0"><sn:metadata><itm-types:time-frame><itm-types:start-time
timestamp="2016-08-01T03:43:18.000Z"/><itm-types:end-time
timestamp=""/></itm-types:time-frame><sn:global-communication-oid>2eb25b2f-
b1b7-4895-ae15-df4afffd94f32585479282862653.local</sn:global-communication-
oid><itm-types:subject content-type="text/plain">Re: Dinner
Friday</itm-types:subject></sn:metadata><sn:participants><sn:participant
participant-role="to"><sn:network-info endpoint-id-type="email"><itm-
types:network>SMTP</itm-types:network><itm-types:endpoint-
id>SPFTRDID@digitbrokerage.com</itm-types:endpoint-id></sn:network-info></
sn:participant></sn:participants><sn:text-events><sn:text-event><itm:text-
object><itm:text-object-style>body</itm:text-object-style><itm:text-content
content-type="text/plain">THIS IS NO JOKE PLEASE SENDTHIS TO ALL OF
CALGARY. THANKS ANGELA /n/n/TODAY IS NATIONAL FUN DAY AT WORK (so get to
work!)/n/nTHANKS FOR ALL YOUR HARD WORK!/n/n/JOHN LAVORATO/n/n</itm:text-
content></itm:text-object></sn:text-event></sn:text-events></sn:snapshot>
```

The sample .xml file for chat data is in the following format:

```
<?xml version='1.0' encoding='UTF-8'?><sn:snapshot xmlns:itm="http://
dig.apc.xmlns.commons.platform.actiance.com/itm/interaction/transcript/1.0"
xmlns:sn="http://dig.apc.xmlns.commons.platform.actiance.com/itm/
interaction/snapshot/1.0" xmlns:itm-types="http://
dig.apc.xmlns.commons.platform.actiance.com/itm/interaction/types/
1.0"><sn:metadata><sn:tenant-id>4627d415-cfd8-41a9-989f-608326d4f672</
sn:tenant-id><sn:cluster-id>chatTranscript</sn:cluster-id><itm-types:time-
frame><itm-types:start-time timestamp="2016-08-01T20:10:30.726Z"/><itm-
types:end-time timestamp="2016-08-01T20:16:22.838Z"/></itm-types:time-
frame><sn:global-communication-oid>adskjakdmkjsldaids13813813.local</
sn:global-communication-oid><sn:global-parent-communication-
oid>Sakjadkamd_928293</sn:global-parent-communication-oid><sn:global-
thread-oid>kjsehdkewh_47364873</sn:global-thread-oid><itm-types:modality
vendor="AOL Inc" classification="uc" type="private"><itm-types:channel>IM</
itm-types:channel><itm-types:network>AIM</itm-types:network></itm-
types:modality><itm-types:subject content-type="text/plain">Conversation
initiated</itm-types:subject><itm-types:groups><itm-types:group>PS</itm-
types:group></itm-types:groups></sn:metadata><sn:demographic/
><sn:transcript-infos><sn:transcript-info><sn:transcript-
```

id>0_78_1441479218265</sn:transcript-id><sn:transcript-timestamp>2016-08-01T20:10:30.726Z</sn:transcript-timestamp><sn:source-endpoint-id>Test</sn:source-endpoint-id><sn:source-endpoint-version>2015</sn:source-endpoint-version><sn:interaction-id>9002</sn:interaction-id><sn:interaction-start-time>2016-08-01T20:10:30.726Z</sn:interaction-start-time><sn:interaction-checksum>0kr2Q767FngA8Ai3nuZXFvQNXKY=</sn:interaction-checksum><sn:interaction-processed-time>2015-09-05T18:53:38.983Z</sn:interaction-processed-time></sn:transcript-info></sn:transcript-infos><sn:participants><sn:participant participant-role="initiator" participant-id="c6c29909a81f89f18c183278b3253b8d"><sn:network-info endpoint-id-type="login"><itm-types:network>AIM</itm-types:network><itm-types:endpoint-id>ADR1517@digitbrokerage.com</itm-types:endpoint-id><itm-types:display-name>ADR1517</itm-types:display-name></sn:network-info><sn:user-info><itm-types:user-id>ADR1517@digitbrokerage.com</itm-types:user-id><itm-types:user-type>internal</itm-types:user-type><itm-types:name><itm-types:first>Joe</itm-types:first><itm-types:last>Driscoll</itm-types:last></itm-types:name><itm-types:affiliation/><itm-types:geo-location/></itm-types:geo-location><itm-types:phone-numbers/></itm-types:phone-numbers/></sn:user-info></sn:participant><sn:participant participant-role="participant" participant-id="d51e4b05c180920db52c91770a60fad3"><sn:network-info endpoint-id-type="login"><itm-types:network>AIM</itm-types:network><itm-types:endpoint-id>XKW1135@digitbrokerage.com</itm-types:endpoint-id><itm-types:display-name>XKW1135</itm-types:display-name></sn:network-info><sn:user-info><itm-types:user-id>XKW1135@digitbrokerage.com</itm-types:user-id><itm-types:user-type>internal</itm-types:user-type><itm-types:name/><itm-types:affiliation/><itm-types:geo-location/><itm-types:geo-location><itm-types:phone-numbers/></sn:user-info></sn:participant></sn:participants><sn:text-events><sn:text-event text-event-id="90003" action="create"><itm:text-object><itm:text-object-id>90003</itm:text-object-id><itm:system-flag>>false</itm:system-flag><itm:text-object-style>normal message</itm:text-object-style><itm:text-object-type>Message</itm:text-object-type><itm:text-object-sub-type/></itm:text-object-sub-type><itm:event-time timestamp="2016-08-01T20:10:30.726Z"/>><itm:participant-id>d51e4b05c180920db52c91770a60fad3</itm:participant-id><itm:text-content content-type="text/plain" indexable="true" user-visible="always visible">Hi, how can you be so irresponsible?</itm:text-content></itm:text-object><sn:creator-participant-id>c6c29909a81f89f18c183278b3253b8d</sn:creator-participant-id><sn:create-time>2016-08-01T20:10:30.726Z</sn:create-time><sn:interaction-checksum>Ni1NBCeiaAz2Rscjmtpm/oH6/EYNPm0QnJLNjC7qxc=</sn:interaction-checksum><sn:snapshot-time>2016-08-01T20:10:30.726Z</sn:snapshot-time><sn:process-time>2016-08-01T20:10:30.726Z</sn:process-time></sn:text-event><sn:text-event text-event-id="90003" action="create"><itm:text-object><itm:text-object-id>90003</itm:text-object-id><itm:system-flag>>false</itm:system-flag><itm:text-object-style>normal message</itm:text-object-style><itm:text-object-type>Message</itm:text-object-type><itm:text-object-sub-type/></itm:text-object-sub-type><itm:event-time timestamp="2016-08-01T20:12:30.326Z"/>><itm:participant-id>c6c29909a81f89f18c183278b3253b8d</itm:participant-id><itm:text-content content-type="text/plain" indexable="true" user-visible="always visible">Hello! What happened?</itm:text-content></itm:text-object><sn:creator-participant-id>c6c29909a81f89f18c183278b3253b8d</sn:creator-participant-id><sn:create-time>2016-08-01T20:12:30.326Z</sn:create-time><sn:interaction-checksum>Ni1NBCeiaAz2Rscjmtpm/oH6/EYNPm0QnJLNjC7qxc=</sn:interaction-

```
checksum<sn:snapshot-time>2016-08-01T20:12:30.326Z</sn:snapshot-
time><sn:process-time>2016-08-01T20:12:30.326Z</sn:process-time></sn:text-
event><sn:text-event text-event-id="90003" action="create"><itm:text-
object><itm:text-object-id>90003</itm:text-object-id><itm:system-
flag>>false</itm:system-flag><itm:text-object-style>normal
message</itm:text-object-style><itm:text-object-type>Message</itm:text-
object-type><itm:text-object-sub-type></itm:text-object-sub-
type><itm:event-time timestamp="2016-08-01T20:13:00.272Z"/
><itm:participant-id>d51e4b05c180920db52c91770a60fad3</itm:participant-
id><itm:text-content content-type="text/plain" indexable="true"
user-visible="always visible">Your service is pathetic!! I have never seen
such a bad customer service. You charge so much for customer service and
hardly provide any service as per the standards</itm:text-content></
itm:text-object><sn:creator-participant-
id>c6c29909a81f89f18c183278b3253b8d</sn:creator-participant-id><sn:create-
time>2016-08-01T20:13:00.272Z</sn:create-time><sn:interaction-
checksum>NiilNBCeiaAz2Rscjmtpm/oH6/EYNPm0QnJLNjC7qxc=</sn:interaction-
checksum><sn:snapshot-time>2016-08-01T20:13:00.272Z</sn:snapshot-
time><sn:process-time>2016-08-01T20:13:00.272Z</sn:process-time></sn:text-
event></sn:text-events><sn:file-events/><sn:policy-events></sn:policy-
events></sn:snapshot>
```

The solution provides a set of services to accept email and chat data.

Publish Email service

This service supports Digest Authentication and publishes email data along with policy code information to the `sifs.email.in` Kafka topic.

REST URL

`https://<host>:<secure_port>/surveillance/data/email`

Header

`Content-Type=text/plain`

`source=Actiance`

REST method

`POST`

Request

`<Policycode>~<Contents of email xml>`

Response

`Success / error message`

Publish Chat service

This service supports Digest Authentication and publishes chat data along with policy code information to the `sifs.chat.in` Kafka topic

REST URL

`https://<host>:<secure_port>/surveillance/data/chat`

Header

`Content-Type=text/plain`

`source=Actiance`

REST method

`POST`

Request

<Policycode>~<Contents of chat xml>

Response

Success / error message

Retrieve Communication service

This service is used primarily to retrieve communication data.

REST URL

https://<host>:<secure_port>/surveillance/data/commevidence/
{communicationId}

REST method

GET

Request

N/A

Response

Payload of communication

Examples of curl commands for publishing email and chat data

```
curl -k -H 'Content-Type: text/plain' -H 'source:Actiance' -X POST --data-binary
@snapshot_002a5d12-4956-4c8b-8fcf-11041eed86d72585479296070162.local.txt --user
actiance1:actpwd1 --digest https://localhost:9443/surveillance/data/email
```

```
curl -k -H 'Content-Type: text/plain' -H 'source:Actiance' -X POST --data-binary
@snapshot_chat_1.txt --user actiance1:actpwd1 --digest
https://localhost:9443/surveillance/data/chat
```

Email data format

The email data that you load into IBM Surveillance Insight for Financial Services must be in CSV format.

The email data must use the following structure:

"USERID", "PCID", "ACTIVITY_TS", "TO_EMAIL", "EMAIL_SUBJECT", "UUID",
"ATTACH", "CC_EMAIL", "BCC_EMAIL", "FROM_EMAIL", "ACTIVITY_CODE",
"ACTIVITY_SIZE", "EMAIL_ATTACH_KEY", "UNIQUEHASH"

USERID

Type: Character

Purpose: User/Trader ID

PCID Type: Character

Purpose: PCID

ACTIVITY_TS

Type: Timestamp

Purpose: Time when the email was sent

TO_EMAIL

Type: Character

Purpose: Email ID of recipient

The recipient can be in the To, CC, or BCC list. If there is more than one entry in the recipient list, create multiple rows in the CSV file, with one row per recipient.

EMAIL_SUBJECT

Type: Character

Purpose: Email subject line

UUID Type: Character

Purpose: Unique identifier

ATTACH

Type: Number

This value is not currently used.

CC_EMAIL

Type: Character

This value is not currently used.

BCC_EMAIL

Type: Character

This value is not currently used.

FROM_EMAIL

Type: Character

Purpose: Email ID of sender

ACTIVITY_CODE

Type: Number

Purpose: Indicates whether the email is inbound (8) or outbound (7)

ACTIVITY_SIZE

Type: Number

Purpose: Size of the email

This value is not currently used.

EMAIL_ATTACH_KEY

Type: Number

Purpose: Unique identifier to the attachment file in the email

This value is not currently used.

UNIQUEHASH

Type: Character

Purpose: Unique hash that is generated for the email body

For example,

```
"AAC0668","PC-9677","2015-03-26
09:58:00.0","Ross.Chandler.Graves@dtaa.com","Re: Annual Report VAR
Disclosures - open items","9ed0b698-9541-4704-a92d-f3893ddb3a1",0,"\N","\
N","AAC0668@digitbrokerage.com",7,3761,0,"d3c2949ad21dff16e3854de826d11cf2833fd234"
```

The email body must be copied to the /var/www/html/emailtext directory.

The emailtext directory contains emails that are represented by the UNIQUEHASH value. For example, if the UNIQUEHASH = d3c2949ad21dff16e3854de826d11cf2833fd234, then the email body is stored in a file that is in the /var/www/html/emailtext/d/3 directory.

The folder structure under /var/www/html/emailtext is derived from first 2 characters of the UNIQUEHASH value.

Voice data format

To create a voice sample, use any voice recording software in which you can record in .wav file that is in uncompressed PCM, 16-bit little endian, 8 kHz sampling, and mono format.

Metadata that is related to the voice sample is captured 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
```

For more information, see “Audio metadata schema” on page 60

The metadata is posted on a Kafka queue for the streams to read and process. A voice metadata application is provided to help post encrypted messages to Kafka. Use the following command to run the application:

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

For more information, see Chapter 9, “Voice surveillance,” on page 63

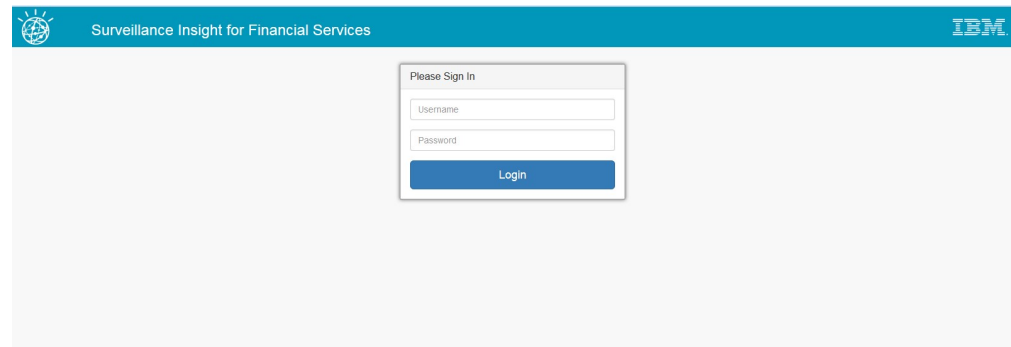
Chapter 3. Use the product

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

The Login page

On the Login page, enter a valid username and password and click the Login button to log into the IBM Surveillance Insight for Financial Services dashboard.

The Login page is available at `http://<hostname>/surveillance`. For the hostname for your installation of the solution, contact your system administrator.



The home page

After you log in, the home page is displayed.

The home page displays the following content:

- **Top Cases** displays a calendar widget. Click a date in the calendar to display the top cases on that date. Dates that do not have any alerts in the system are disabled by default. Clicking a case, displays a page with detailed information about the case.
- **Live Monitoring & Replay** shows a snapshot of the trader and ticker relationship graph
- **Know Your Trades** links to the Know Your Trades page
- **Know Your Trader** links to the Know Your Trader page
- **Trade Analysis** shows the number of anomaly alerts from structured data analysis, and it links to the Trade Analysis page
- **Communication Analysis** shows the number of anomaly alerts from unstructured data analysis, and it links to the email communication analysis page



Trade analytics


The Trade analytics component monitors the activity of traders to detect market abuse activities.

The solution includes prebuilt models to detect pump-and-dump and spoofing scenarios. In addition, the component offers reusable operators such as a bulk order detector to detect base alerts.

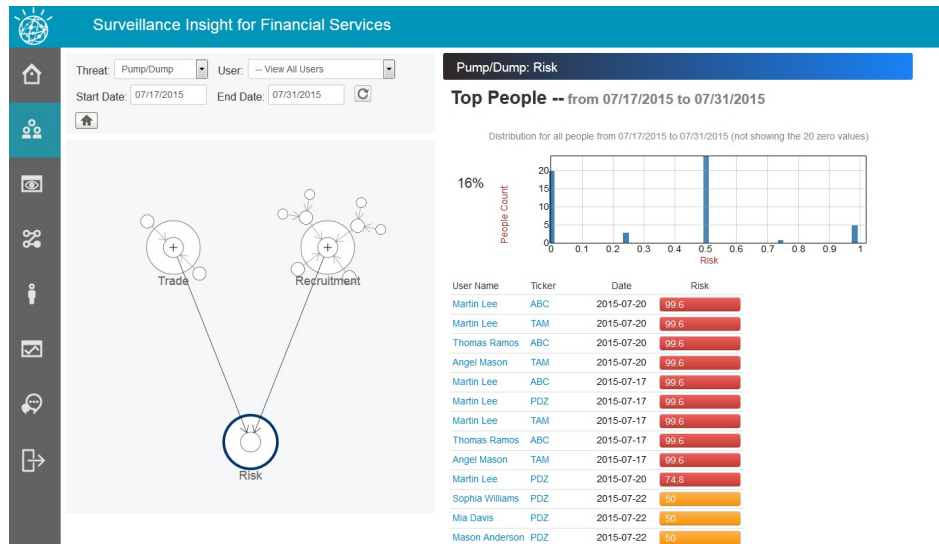
The Pump and Dump Reasoning page

This Pump and Dump Reasoning page displays the reasoning graph and the date wise scores for the selected trader.

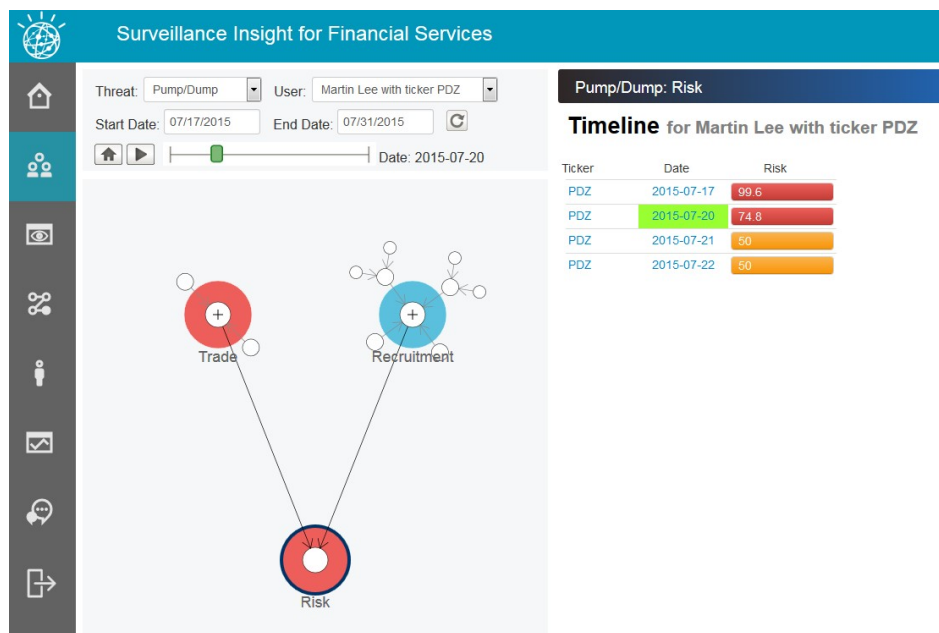
The page appears when a ticker bar is selected from the Top Cases widget in the home page. Selecting the specific ticker or date in the PumpDump Risk section, updates the reasoning graph. The page can also be accessed by clicking the

Reasoning icon  on the navigation bar. When you access the page from the Reasoning icon, you need to select a specific date range in the date widget and click the **Refresh** button to display the reasoning graph.

Alternatively, you can use the date slider to set a specific date and then click the **Play** button to see the reasoning graph change over the selected time period.

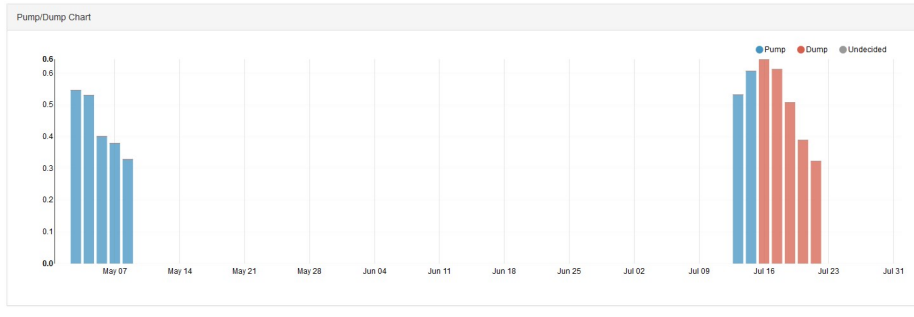


If you click a specific trader from the list of traders, the following page displays the details of the tickers and the dates related to that trader.

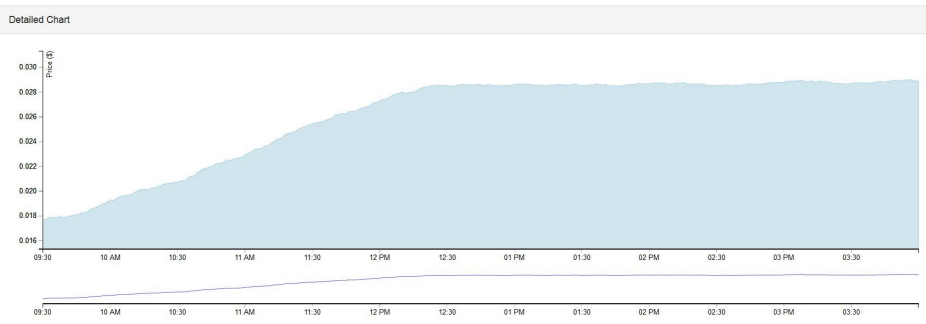


Clicking the ticker displays the Details page with the ticker details and the Pump and Dump chart for that ticker, as shown in the following two screen captures.

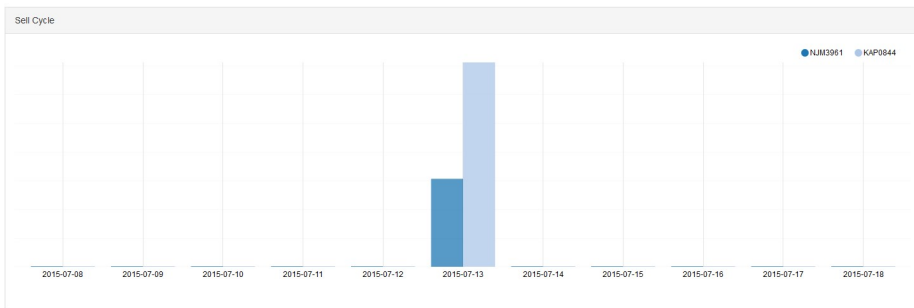




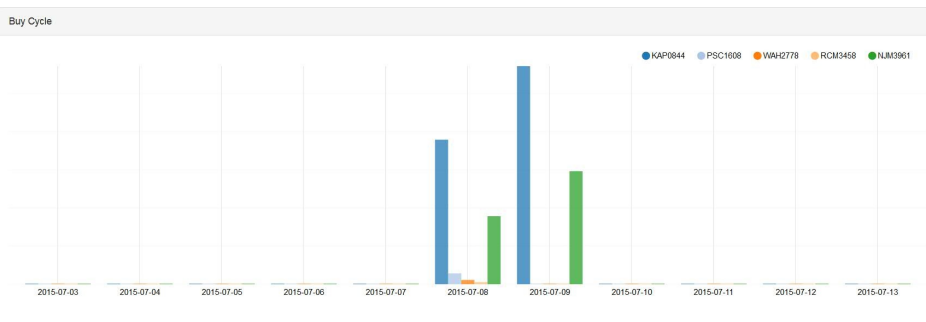
Clicking a specific bar in the Pump and Dump chart displays a detailed price chart for that date as shown in the following screen capture.



You can view the sell cycle.



And, you can view the buy cycle.



The Spoofing Charts page

If any spoofing alerts are detected by the system, they are displayed in the Top Cases widget on the home page. Clicking the alert takes you to the Spoofing charts page, which shows the Spoofing chart and the Profit chart.

The Spoofing Chart shows the spoofing pattern that was detected in terms of orders, cancellations, and executions that were completed by a trader. It also shows the change in the ask and bid prices of the quotes and the order prices that the trader used. The chart indicates the orders and cancellations that were made by the trader over time. It also shows the partial executions that occurred before the orders were canceled, and the executions made by the trader after the bid price was spoofed to a higher price.

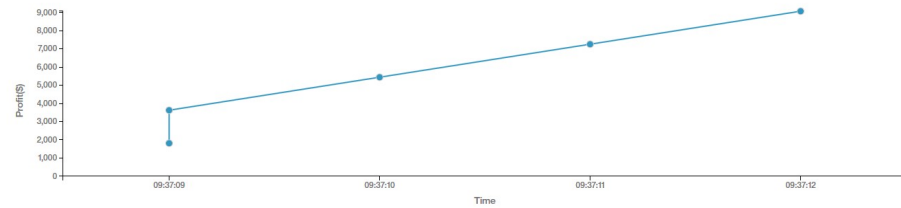
Orders and Executions for Trader ANGEL MASON on symbol KO for Date : 2015-05-08

Spoofing chart



The profit chart shows the trader's profit over time.

Profit chart

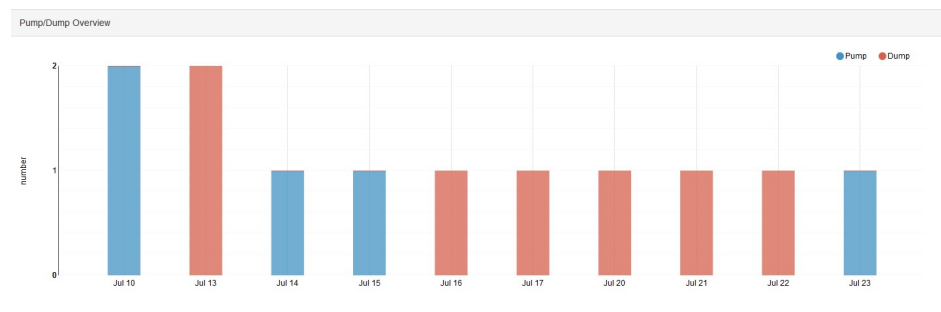


Trade Analysis page

The Trade Analysis page is linked from the Trade Analysis widget in the home page. It contains a Pump and Dump Overview, Activity Overview, Anomaly chart, and Trade chart,

Pump and Dump Overview chart

In the Pump and Dump Overview bar graph, the height of the bar indicates the daily score and the color of the bar indicates either pump or dump.



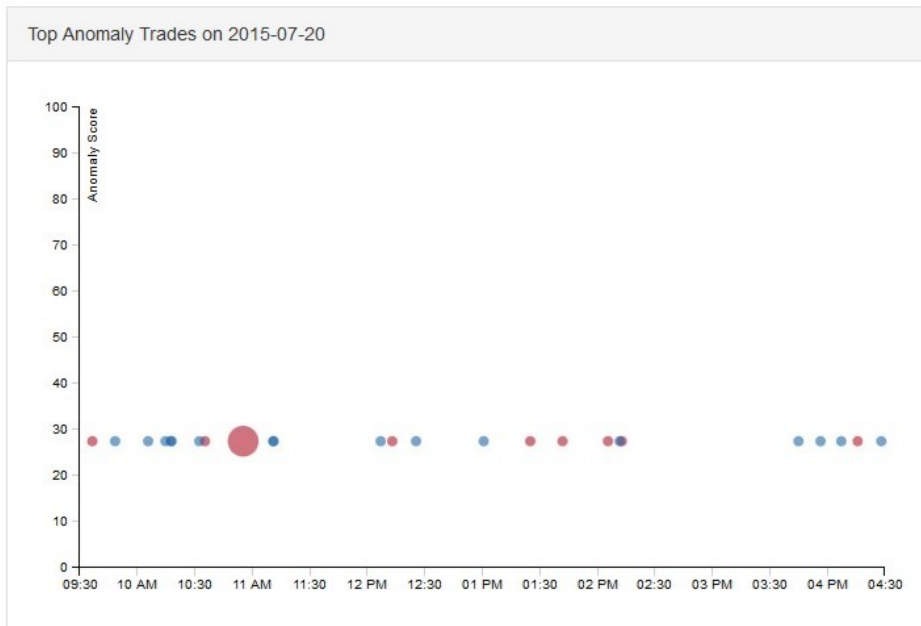
Activity Overview table

The Activity Overview table is displayed when you click a bar in the Pump and Dump Overview chart. It shows information about the ticker for the selected date. Clicking the Details column shows further details about the ticker's price variation and pump dump stages over the selected period.

Activity Overview on 2015-07-20									
Ticker	52-Week High	52-Week Low	Description	Stage	Industry Code	Closing Price	Total Volume	Score	Details
PDZ	0.696	0.328	PDZ Corporation	Dump	Healthcare	0.652	168781780	50.86	

Anomaly chart

The Anomaly chart is displayed when you click a bar in the Pump and Dump Overview chart. It indicates anomalous behavior by traders on the timeline for the day. The size of the bubble indicates the size of the trade.



Trade chart

The trade chart is displayed when a bubble in the Anomaly Chart is clicked. It plots the price of the ticker over time for the day, indicating the anomalous trader that was clicked in the anomaly chart.



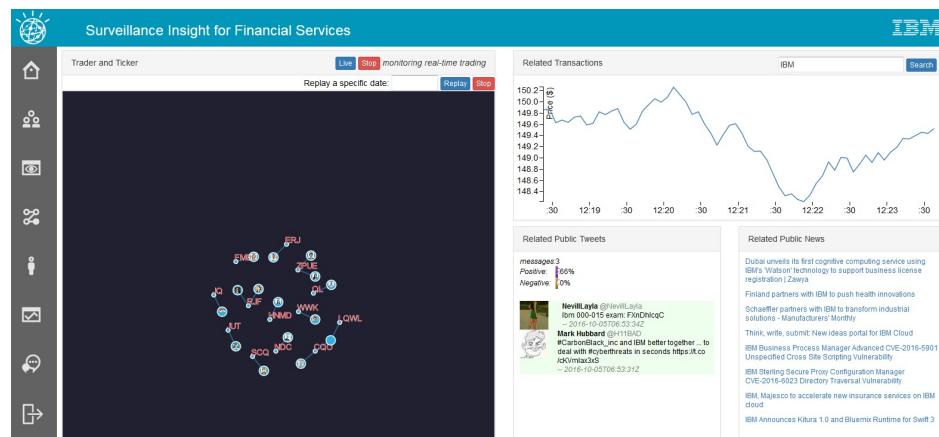
Live Monitoring page

The Live Monitoring page shows a live visual representation of the relationships between a ticker and the trades for that ticker. It is displayed when you click the Transaction Monitoring and Replay widget.

The Live Monitoring page displays the following information:

- **Trader – Ticker graph** shows the relationships between a ticker and the traders for that ticker. It updates over time. The live update can be stopped or played.
- **Related Transaction** shows a live line graph of the price of the ticker over time
- **Related Public Tweets** shows a feed of the tweets related to the ticker that is being viewed
- **Related Public News** shows public news feeds about the ticker that is being viewed

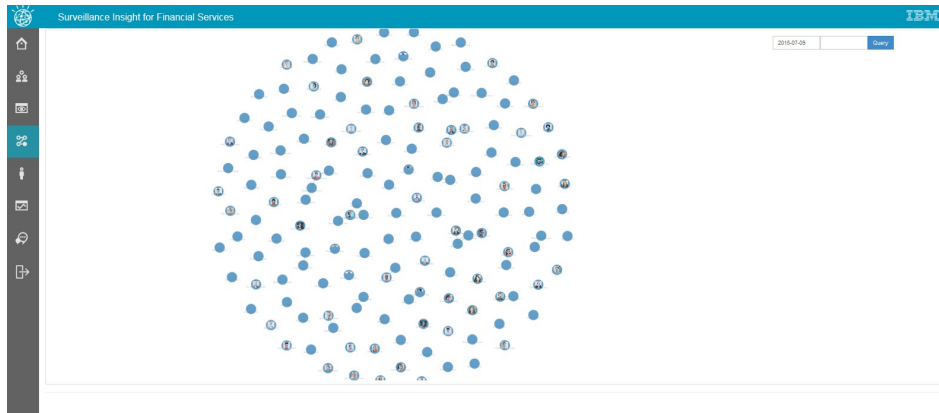
From the Live Monitoring page, you can also search for tickers and refresh the content with the searched ticker.



Know Your Trades page

The Know Your Trades page provides a visual representation of the trades that occurred between different traders.

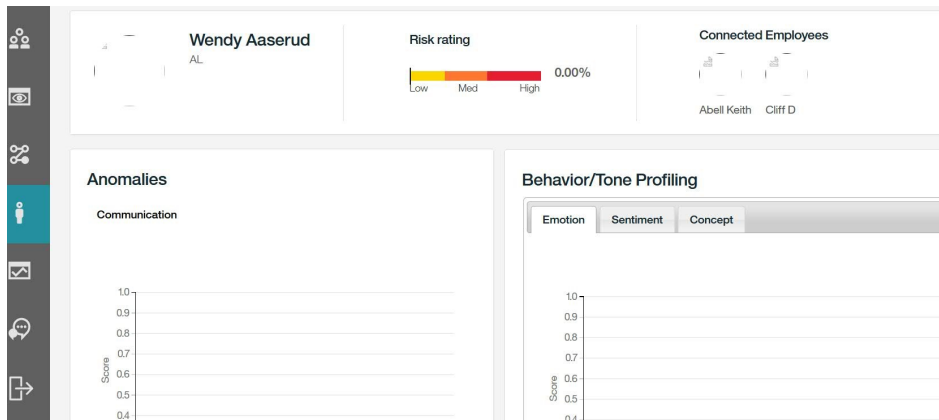
Click the trader node to zoom in. Hover your mouse pointer on the trader node to display the analytic degrees for the trader along with the trader ID.



Know Your Employee page

The Know Your Employee page contains a list of employee names.

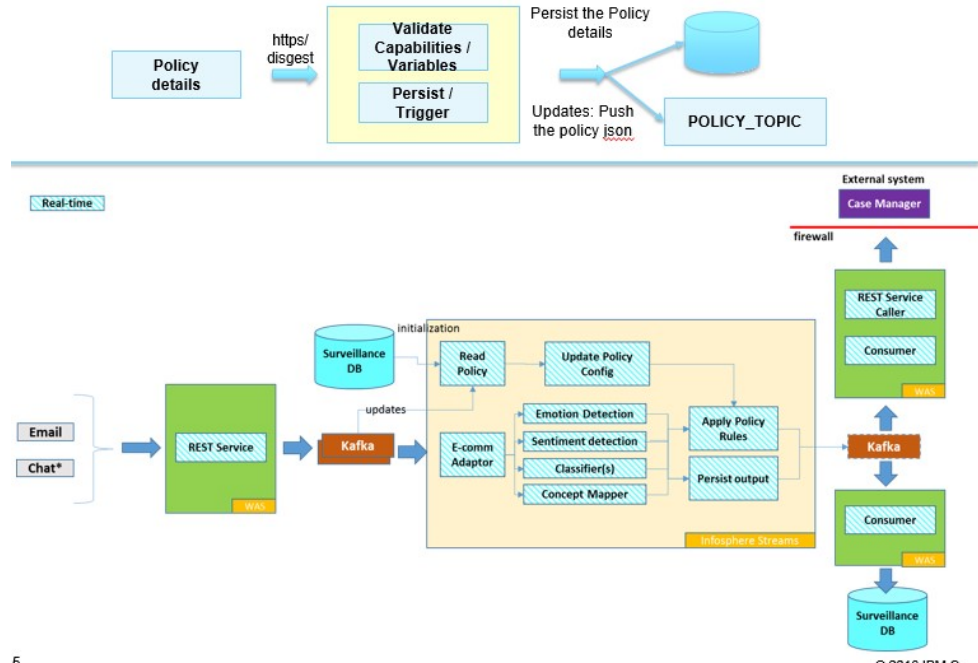
When you select an employee, the following information is displayed:



E-communication analytics

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

The following diagram shows the architecture of the E-Communication analytics that are used in the solution.



The Communication Alerts page shows the different alerts that are generated by the system based on the analytics that are performed on unstructured communication data such as voice, email, and chat messages.

As shown in the following screen capture, the results show up on searching for “*”. Results are displayed in a table.

Communication Search

Q

Emotions Clear

- Anger(85)
- Sad(112)
- Joy(111)
- Disgust(98)
- Fear(106)

Sentiment

- Positive(57)
- Negative(37)

Concepts

- Recruit victims(52)
- Recruit conspirators(54)

Classification

- Trading(0)
- Non-trading(0)
- Confidential(0)
- Non-confidential(0)
- Business(0)
- Personal(0)
- News(0)
- Announcement(0)
- Promotional(0)

Filter By ^

Showing 1-10 of 999

COMMUNICATION TYPE	DATE	TIME	FROM	TO	
voice	2016-07-10	09:10AM	NA	NA	Show details
voice	2016-02-02	02:10PM	NA	NA	Show details
NA	NA	NA	NA	NA	Show details
NA	NA	NA	NA	NA	Show details
voice	2016-04-05	09:45AM	NA	NA	Show details
voice	2016-03-13	11:10AM	NA	NA	Show details
voice	2016-07-11	10:21AM	NA	NA	Show details
EMAIL	2016-09-08	09:10AM	a@af.com	c@d.com,c@g.com	Show details
Chat	2016-09-02	02:29PM	a@af.com	c@d.com,c@g.com	Show details
VOICE	2016-04-08	09:34AM	a@af.com	c@d.com,c@g.com	Show details

< 1 2 3 4 5 >

When you click the Show Details link, the emotions that are detected in the data are displayed as shown in the following screen capture. The size of a word indicates the relative number of times that an emotion was detected.



Running a pump and dump analysis

Use the following steps to run a pump and dump analysis in IBM Surveillance Insight for Financial Services.

For more information about the input data files, see Chapter 8, “Data schemas,” on page 53.

Before you begin

Ensure that the `initpositions.csv` file is present in the data folder for PositionSummary. This file contains the initial positions for the traders based on the ticker.

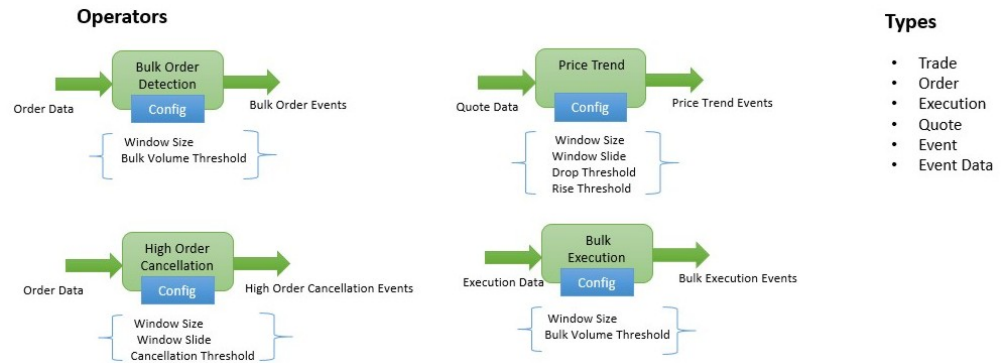
Procedure

1. Copy the `trade.csv` and `execution.csv` files for day 1 to the data directory for MarketDataLoader.
You must wait for the `<date>_trade_complete.csv` file to be created in the data directory for TradeSummary and the `<date>_executions_complete.csv` file to be created in the data directory for PositionSummary. These two files indicate that the calculations are completed for the day.
2. Copy the end of day file for day 1 in the data directory for MarketDataLoader. The end of day file updates the following graphs with the analysis results:
 - Trade summary data in the `ticker_summary_<symbol>` and `movingaverage` graphs
 - Position summary data in the `_position` and `Top5Traders` graphs
 - End of day records in the `symbol_eod` graph
 - Pump and dump scores in the `pumpdump` and `<date>_PD` graphs
 - Trader scores in `score` and `<date>_score` graph
3. Repeat steps 1 - 2 for each subsequent day.

Chapter 4. Trade Surveillance Toolkit

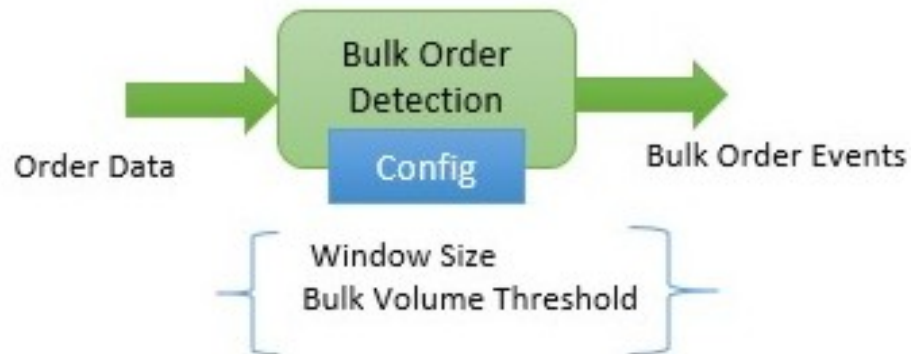
The Trade Surveillance Toolkit helps the solution developers to focus on specific use case development.

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 Chapter 8, “Data schemas,” on page 53.

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

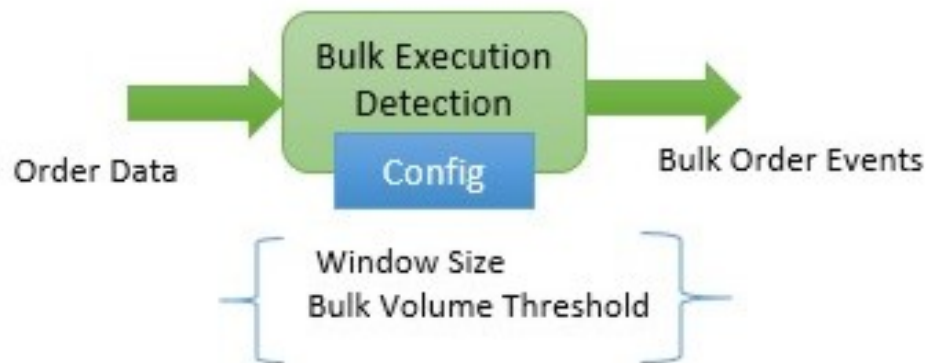
Window Size: time in seconds for collecting data to analyze.

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

Refer to Chapter 2, "Load data," on page 9 for instructions on loading market data.

When data is loaded into the solution, the Pump-and-Dump rules are triggered and the following graphs in the graph database are updated:

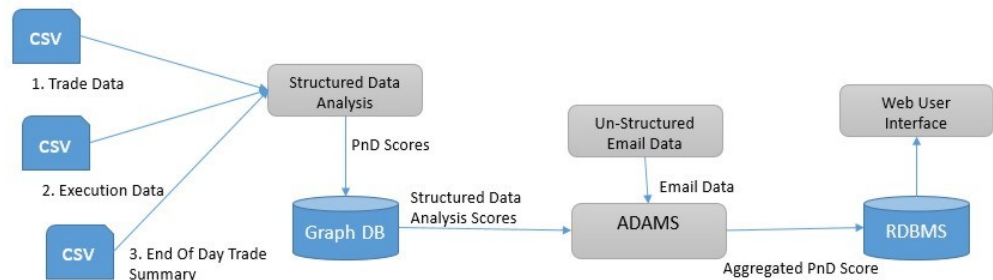
- Trade summary data in ticker_summary_<symbol> and movingaverage graphs
- Position summary data in _position and Top5Traders graphs
- EOD record in symbol_eod graph
- Pump-and-Dump scores in pumpdump and <date>_PD graph
- Trader scores in score and <date>_score graph.

Pump-and-Dump User Interface

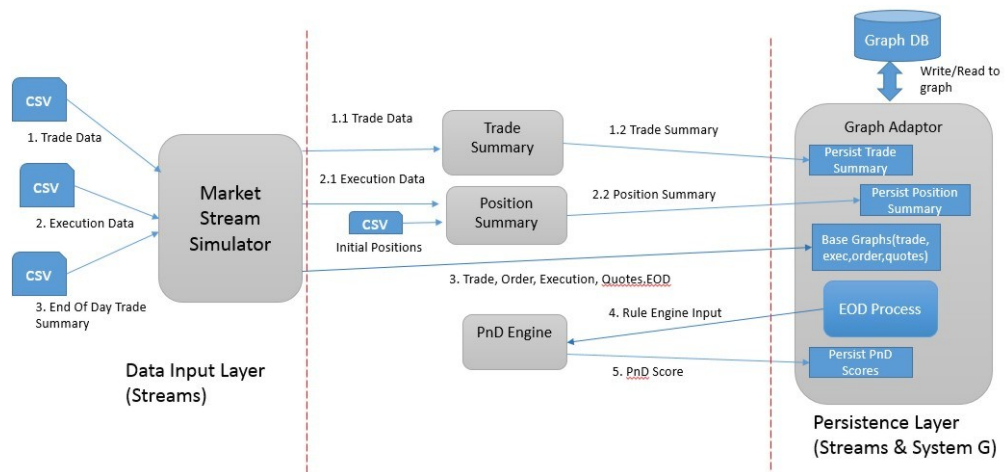
Refer to Chapter 3, “Use the product,” on page 19 for information on how to use the web-based user interface for pump-and-dump analysis.

End-to-end data flow view

Understanding the solution's pump-and-dump design helps solution developers to create similar implementations. The following diagram shows, the end-to-end view of the Pump-and-Dump analysis.



The structured analysis score is combined with the unstructured analysis scores, and the resulting alerts are persisted to the tables in a relational database. The User Interface components pick the results from the RDBMS and Graph DB and display them on the web interface. The structured analysis portion is shown in the following diagram:



- The Market Data Loader feeds the .csv content to the graph database and also to the Trade and Position Summary streams.

- Summaries are computed by the Trade and Position Summary streams on a day's data. After that, the End of Day processing begins in the EOD Process stream.
- The Summaries are read from the graph DB and the pump-and-dump rules are invoked as part of the EOD Process.
- The scores that are returned by the pump-and-dump (PnD) engine are persisted to the graph database.

Spoofer detection use case

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

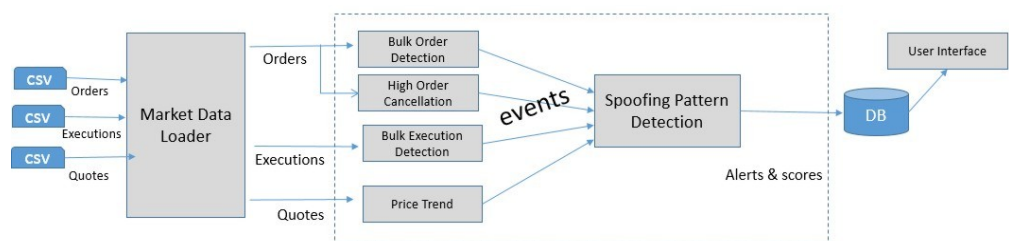
A spoofer is a trader who creates a series of bulk buy or sell orders with increasing bid or decreasing ask prices with the intention of misleading the buyers and sellers in a direction that results in a profit for the spoofer. The spoofer cancels the bulk orders before they are completed and then sells 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 orders, executions, and quotes data to detect the spoofing pattern. Refer to Chapter 2, "Load data," on page 9 for instructions on loading market data. These steps trigger the Spoofing rules and update the alert and score tables in the database.

End-to-end data flow view

The spoofing use case uses 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 following diagram shows the high-level design:



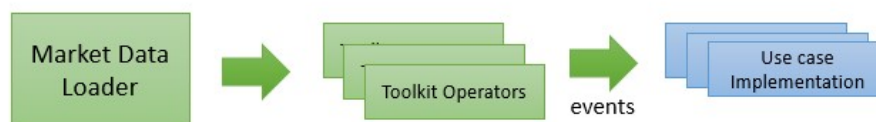
Guidelines for developing new models

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

Understand the event-based approach to performing trading analytics

The Trade Surveillance Toolkit generates events based on stock market data. The toolkit defines a basic event type that can be extended 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. These operators generate different types of events based on their analysis. A benefit of the event-based model, is that it allows the specific use case implementations to delegate the basic functions to the toolkit and focus on only events that are relevant. The model allows the events to be generated once and reused by all interested use cases. It also drastically reduces the volume of data that the use cases need to work with.



Identify the data types and analytics relevant to the use Case

Identify the data that is relevant to the context and the analytics that need to be applied to that data. The analytic measures are then used to identify the events that are of interest.

Identify the Trading Surveillance Insights Toolkit contents for reuse

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

Design the stream flows by using the operators in the toolkit

This step is specific to the use case that is being implemented. During this step the following actions occur:

- The placement of the SI Toolkit operators in the context of the larger solution is identified.
- The configuration parameters values for the different operators are identified.
- Additional data types and operators that are not present in the toolkit are designed.

Implement and test the solution.

Chapter 5. 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;

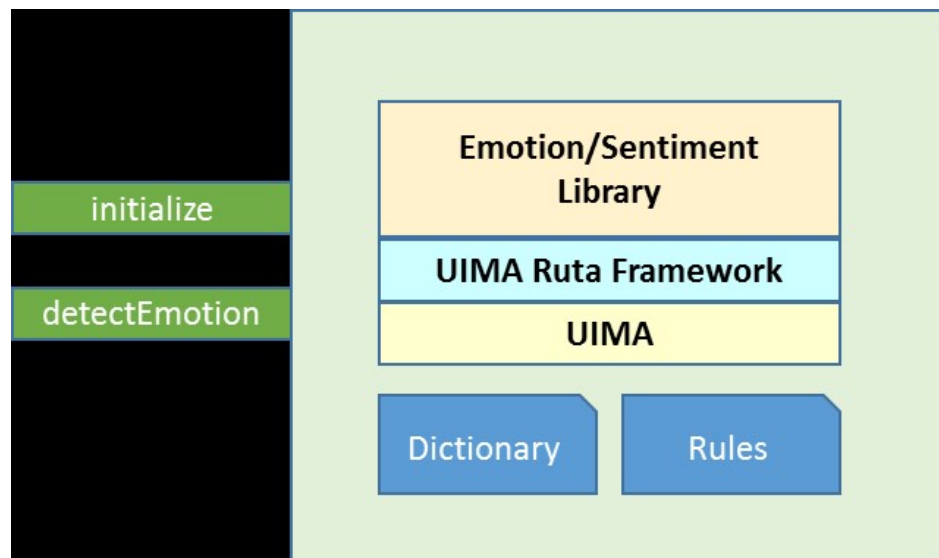
MARKFAST(Anger, anger_dict, true);
MARKFAST(Joy, joy_dict, true);
# Same for other emotions
```

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.



Definitions

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

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';

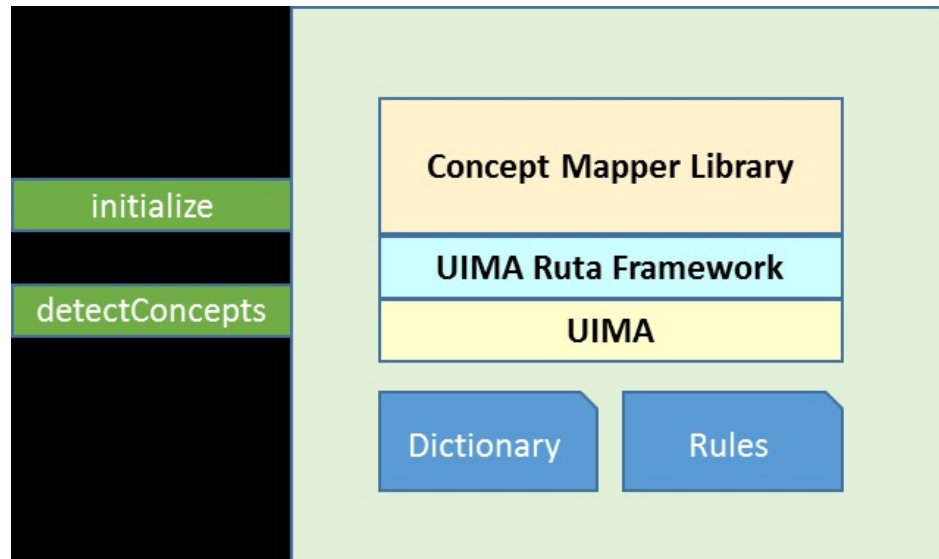
# Type definitions
DECLARE Ticker;
DECLARE RecruitConspirators;
DECLARE RecruitVictims;
DECLARE Negative;

# 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.



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

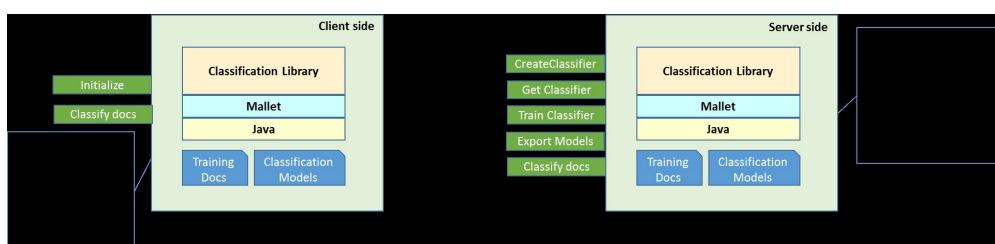
How the library works

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.



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 2. Server-side library

Method	URL	Input	Output
POST	/text/v1/classifiers	JSON 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 3. 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 4. 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 5. 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 6. 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 7. 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://localhost:9080/analytics/text/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 6. Policy

The solution provides a feature to define policy. Policy can be defined for a group of users or for all employees across the organization.

The following is a sample JSON policy definition.

```
{
  "policy": {
    "name": "Policy 1",
    "code": "POL1",
    "capability": [{
      "name": "sentiment",
      "variables": [
        {"name": "score"}, {"name": "type"} ]
    }, {
      "name": "emotions",
      "variables": [
        {"name": "anger"}, {"name": "sad"} ]
    }
  ],
  "rule": "sentiment.score > 0.5 && sentiment.type == 'negative' && emotions.anger > 0.5 || emotions.sad > 0.5",
  "channel_scope": "e-comm"
}
```

For information about the elements in the Policy schema, see “Policy schema” on page 61

Capabilities

The solution provides the following capabilities that are related to policy execution:

- Sentiment Detection
- Emotion Detection
- Classifier
- Concept Mapper

Table 8. Capabilities

CapabilityName	CapabilityAttrName	Remarks
Sentiment	type	Type of sentiment e.g., negative or positive
Sentiment	score	Sentiment score
Emotions	sad	Sad score
Emotions	anger	Anger score
Concepts	tickers	List of tickers that are found in the incoming data
Concepts	recruitvictims	If there is any condition of recruit victims in the incoming data
Concepts	recruitconspirators	If there is any condition of recruit conspirators in the incoming data

Table 8. Capabilities (continued)

CapabilityName	CapabilityAttrName	Remarks
Classes	confidential.class_name	If the incoming data belongs to confidential class
Classes	confidential.confidence	Confidence value that incoming data belongs to confidential class
Classes	non-confidential.class_name	If the incoming data belongs to non-confidential class
Classes	non-confidential.confidence	Confidence value that incoming data belongs to non-confidential class
Classes	top_class	Incoming data belongs to which class i.e. confidential or non-confidential

From the capabilities, the solution derives key performance indicators (KPIs), such as anomaly scores, which are primarily displayed on the Know Your Employee page. All the KPIs are computed for all employees at a daily level. The following list shows the types of key performance indicators:

- Inbound Volume KPI
- Outbound Volume KPI
- Anger Volume KPI
- Unhappy Volume KPI
- Negativity Sentiment Volume KPI
- Confidential Outbound Volume KPI

Alert services and policy execution

The solution provides a service for registering cases in the Case Management system. The service accepts the case message in JSON format. It is invoked as part of the policy execution and is not invoked directly by users.

The solution provides a feature to execute a policy defined by the user of the incoming data that originates from email, chat, or voice messages. The policy execution is realized using Infosphere Streams. As part of Policy execution, the following steps are completed:

1. The system loads all policies registered in the system as part of the job start up. Any change in policy after the jobs are submitted, requires a resubmission of the jobs
2. The system loads the master list of capabilities that are supported by the system.
3. When system receives a request to execute the policy, it parses the policy and executes the capabilities, which are defined in the policy and also executes the capabilities, which are not defined in the policy but are marked as default in the CAPABILITY_ATTR database table.
4. After all the capabilities are executed, the system evaluates the rule expression defined as part of the policy. If the rule evaluates to True, then the system raises an alert and registers a case into the Case Management system. If the rule evaluates to False, no further action is taken.

5. As part of the capability execution, the results of the capabilities are persisted in various tables.

Defining rules

As part of policy, a rule expression is defined and this rule expression is evaluated by the Policy Execution engine. The elements that can be used in rule expression are listed in the following table.

Table 9. Elements used in rule expressions

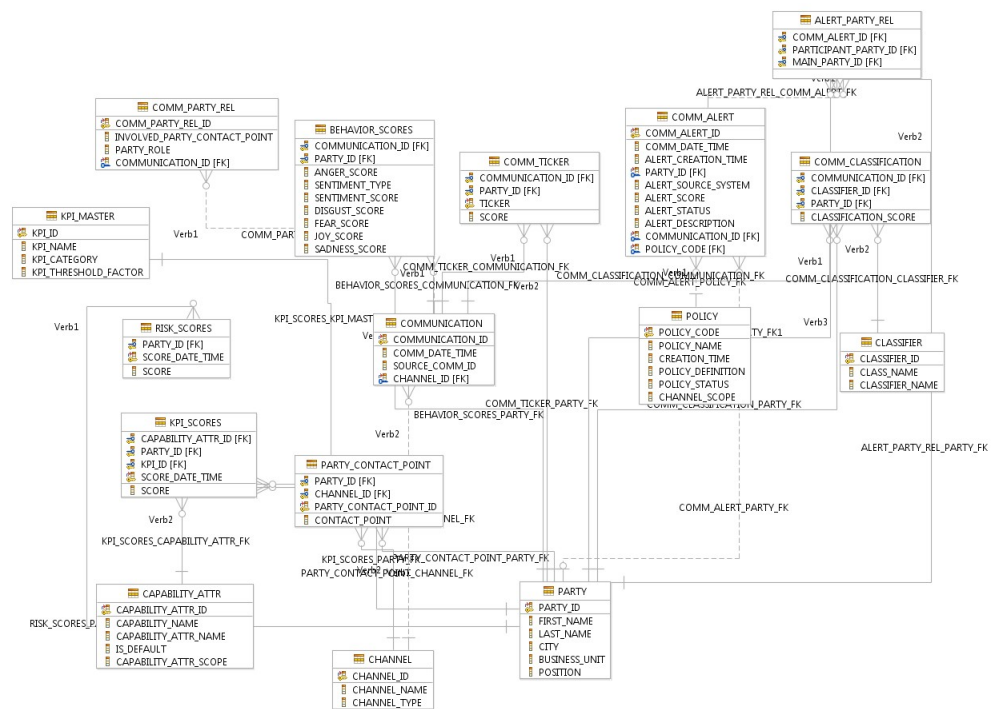
CapabilityName	CapabilityAttrName	RuleExpression	Data Type	Remarks
sentiment	Type	sentiment.type	character	
sentiment	Score	sentiment.score	number	
emotions	Sad	emotions.sad	number	
emotions	Anger	emotions.anger	number	
concepts	Tickers	concepts.tickers	character	
concepts	recruitvictims	concepts. recruitvictims	boolean	
concepts	recruitconspirators	concepts. recruitconspirators	boolean	
classes	confidential.class_name	classes. confidential.class_name	character	
classes	confidential.confidence	classes. confidential.confidence	number	
classes	non-confidential.class_name	classes. non-confidential.class_name	character	
classes	non-confidential.confidence	classes. non-confidential.confidence	number	
classes	top_class	classes.top_class	character	
outbound	Score	employee.inbound_score	number	Inbound Anomaly score
inbound	Score	employee.outbound_score	number	Outbound Anomaly score
emotions	Anger	employee.emotions_anger_score	number	Emotion Anger Anomaly score
emotions	Sad	employee.emotions_sad_score	number	Emotion Unhappy Anomaly score
sentiment	Score	employee.sentiment_score	number	Negativity Sentiment Anomaly score
confidential	Score	employee.confidential_score	number	Confidential Anomaly score

Chapter 7. E-comm schema and persistence overview

This section information on the e-comm schema, persistence, and indexing for IBM Surveillance Insight for Financial Services.

Schema Overview

The schema for the solution provides entities to store the communication data and the results of various analytics and KPI computations. Tables are used to store master data that the system needs such as party details (PARTY table), party contact point details (PARTY_CONTACT_POINT table), capabilities supported by system (CAPABILITIES table), data channel (CHANNEL table), different types of KPI that the system computes (KPI_MASTER table), and a list of classifiers that are supported (CLASSIFIER). An overview of the schema is shown in the following diagram.



Data persistence

E-Comm data is processed by the solution in two steps. First, email and chat data is parsed by the solution and converted into a communication tuple. Then the communication tuple object is processed by the policy execution job. As part of the processing, data is persisted into the database in various tables:

- All communication data is identified by Communication Id and persisted in the COMMUNICATION table and COMM_PARTY_REL table.
- If the incoming data belongs to a party which is registered in the Surveillance system, then policy execution will initiate further analysis.
- With policy execution various capabilities are invoked and the results of those capabilities are stored in the following tables:

- BEHAVIOR_SCORES – Results of Emotion and Sentiment capability
- COMM_CLASSIFICATION – Results of Classifier capability and concept mapper capability
- COMM_TICKER – Results of Concept Mapper library for ticker identification
- With policy execution, rule expression is evaluated. If a rule is evaluated as True, then an alert is persisted in the COMM_ALERT and ALERTY_PART_REL tables.
- With policy execution, various key performance indicators are computed at a daily level and the results are stored in the KPI_SCORES table.
- A consolidated risk score is computed from all six KPI scores and is stored in the RISK_SCORES table

Chapter 8. Data schemas

This section shows the data schemas for input files for IBM Surveillance Insight for Financial Services.

Ticker price schema

symbol,datetime,price

Table 10. 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, settlType, settlDate, Currency, currencyFXRate, execType, trdType, matchType, Side, orderQty, Price, exchangeCode, refQuoteId, refOrderId

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

Table 11. 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

Table 11. Execution schema (continued)

Field name	Field type	Description
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 Currency to SettlCurrency
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	

Table 11. Execution schema (continued)

Field name	Field type	Description
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 FIXwiki (<http://fixwiki.org/fixwiki/NewOrderSingle/FIX.5.0SP2%2B>)

Table 12. 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
settlType	String	Indicates order settlement period. If present, SettDate overrides this field. If both SettType and SettDate are omitted, the default for SettType 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

Table 12. Order schema (continued)

Field name	Field type	Description
currencyFXRate	Float	The exchange rate that is used to calculate the SettlCurrAmt from Currencyto 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 FIXwiki (<http://fixwiki.org/fixwiki/Quote/FIX.5.0>)

Table 13. 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 Currencyto 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	

Table 13. Quote schema (continued)

Field name	Field type	Description
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	
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 14. 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 15. 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
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 16. 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 17. 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 18. 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 ;

Table 18. Audio metadata schema (continued)

Field name	Field type	Description
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

Policy schema

Table 19. Policy schema

Element name	Purpose	Comments
name	Name of policy	
code	Policy Code	Should be unique across all policies
capability	Different capabilities to be executed as part of policy	The list of capabilities that a policy can have
rule	Expression to be evaluated	Users can write a free-form expression while defining a policy. The expression can be written on the list of capabilities supported by the solution. While writing expressions, ensure that for number comparisons do not use the plus sign (+) for positive numbers. Instead just write the number, e.g., (emotions.sad > 0.5)
channel_scope	Scope of policy	

Communication object tuple

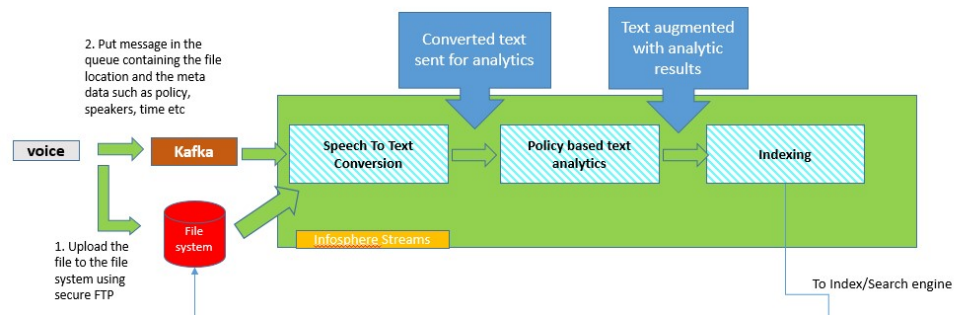
Table 20. Communication object tuple

Element name	Data type	Purpose
uniqueHash	character	Unique hash value of source communication Id
trader_id	character	Party id of employee
commID	character	email id or chat id
activity_ts	character	Date time of the communication
emailBody	character	Email of Chat content
globalCommId	character	Source communication id
participants	map	Map of email id and its corresponding party id
direction	character	Direction of email, inbound or outbound
outboundLog	character	Outbound log flag
policyId	character	Policy Code
communicationId	character	Communication Id

Chapter 9. Voice surveillance

In a trading house, traders use a variety of voice communication channels to interact with clients, brokers and fellow traders. Employees of an organization may talk to family, friends, and people in rival organizations. To mitigate the risks from inappropriate communication, unauthorized trading, and inappropriate trading activities and to satisfy regulatory requirements, there is a need to monitor voice communications.

Voice surveillance depends on the Speech2Text toolkit provided by IBM Streams to convert speech audio into text. The toolkit contains default model and configuration files that enables it to generate text that corresponds to the speech that is processed.



The toolkit accepts audio files in .wav format, which must be in uncompressed PCM, 16-bit little endian, 8 kHz sampling, and mono format. To convert to this format from a .wav file, use the following ffmpeg command:

```
$ ffmpeg -i wrongFormat.wav -ac 1 -ar 8000 correctFormat.wav
```

Voice Adapter is an IBM Streams component that processes the .wav files and aggregates the utterances to be processed by downstream analytics. The adapter employs an instance of the Speech2Text operator for text conversion. The audio .wav files are expected to be placed in a designated directory. This adapter listens on a sifs.voice.in Kafka queue. The adapter initiates its processing only when it's triggered by a message on this queue. The message is in .csv format and must adhere to 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
```

For more information, see “Audio metadata schema” on page 60.

The converted text is then processed through a policy-based analytics job.

Indexing and search

The solution provides search capabilities that enables administrators to search for any kind of communication email, chat, or voice message, based on keywords or other attributes. Indexing is performed on the content of each communication

irrespective of channels as well as on results of the analytics. In the communication search page, you can search for keywords that are run against a Solr Index database. Keyword Search can be done in the following ways:

- To search for communications containing any words from a list, simply list the words.

Communication Search

hectic taxing volume

- To search for documents containing all words in the criteria, use the AND operator.

Communication Search

hectic AND taxing AND volume

Appendix. 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 (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 ” at www.ibm.com/legal/copytrade.shtml.

Index

A

accessibility 65
activity overview table 23
alert services 48
anomaly chart 23
architecture 5
audio metadata schema 60

B

bulk execution detection 29
bulk order detection 29
buy cycle 20

C

classifier library 42
cognitive analysis and reasoning component 5
communication alerts page 26
communication object tuple 62
compliance workbench 5
components of the solution 4
concept mapper 37
concept mapper library 39

D

data ingestion 5
 e-comm data 11
 voice data 17
data loading 9
data store 5
document classifier 37

E

e-comm data ingestion 11
e-comm schema 51
e-communication analytics 26
email data
 configuring to load 10
email data format 15
emotion detection 37, 47
emotion detection library 37
end of day schema 59
event data schema 60
event schema 60
execution schema 53

G

guidelines for new models 34

H

high order cancellation 29
home page 19

I

IBM Streams 63
indexing and search 63
introduction v

J

JSON policy definition 47

K

know your employee page 26
know your trades page 26

L

live monitoring page 25
load data 9
 structured data 9
login page 19

M

models
 guidelines 34

N

natural language libraries 37
 classifier 42
 concept mapper 39
 emotion detection 37

O

order schema 55
overview 1

P

pages
 home 19
 know your employee 26
 know your trades 26
 live monitoring 25
 login 19
 pump and dump reasoning 20
 spoofing charts 23
 trade analysis 23
persistence 51
policy 47
policy execution 48
policy schema 61
price trend 29
pump and dump 28
 reasoning page 20
 use case 32

pump and dump overview chart 23

Q

quote schema 56

R

real-time analytics 5
related public news 25
related public tweets graph 25
related transactions graph 25
running pump and dump job 28

S

schema 51
schemas
 audio metadata 60
 data 53
 end of day 59
 event 60
 event data 60
 execution 53
 order 55
 policy 61
 quote 56
 ticker price 53
 trade 58
sell cycle 20
sentiment detection 47

solution architecture 5
speech2text toolkit 63
spoofing
 use case 34
spoofing charts page 23
structured data 9

T

ticker price schema 53
trade analysis page 23
trade analytics 20
trade chart 23
trade schema 58
trade surveillance toolkit 29
trader-ticker graph 25
tuple 62

U

unstructured data
 configuring to load 10
use case
 pump and dump 32
 spoofing 34

V

voice data ingestion 17
voice surveillance 63