SpinBoard Product Overview

Introduction to SpinBoard

What is SpinBoard?

- ***SpinBoard*** is a highly customizable, Web based, low-code, aggregation and visualization platform for heterogeneous data.

- ***SpinBoard*** is built with two main concepts in mind:
  - Provide the user with an extremely intuitive, self-configurable interface
  - Easily allow the integration of huge amounts of data, coming from heterogeneous sources

Introduction to SpinBoard

Widgets as the main concept for the users

- ***SpinBoard*** provides a set of widgets to display data
- Every user can arrange the widgets as preferred:
  - the system automatically preserves the interface status and layout
  - two users can display the same data in a completely different way, according to their preference.

- ***SpinBoard*** is designed to integrate and display streams of data and provides a set of ready to use data integration flows
  - when the integration of a new stream is needed, little or no java code is needed

SpinBoard Goals

Born to meet the customer needs

SpinBoard was born to meet the specific customer need to monitor and manage a complex system with as low as possible effort:

- Providing a real-time system status overview
- Providing support information for investigation and analysis
- Enabling pro-active issues management
- Integrating existing instruments and software

All the instruments already in place are providing needed information, but in a fragmented way, thus resulting in high investigations times and a lot of skills required for the job.

Functionalities Comparison

SpinBoard compared to other monitoring tools

Following a brief list of the most used / necessary functionalities

Functionality	Geneos	Splunk	SpinBoard	Notes
gRPC Connector	N	Y	Y	
ZMQ Connector	N	Y	Y	
Aeron Connector	N	N	Y	
Custom Localization	N	N	Y	SpinBoard supports custom translations
Custom Data Segregation	N	N	Y	
Hierarchical Grids	N	N	Y	
Grids filtering / sorting	N	Y	Y	Splunk requires query modification
Cloud Ready	N	Y	Y	
Task Execution	Y	N	Y	
Oracle/sql	N	Y	Y	Splunk lacks custom data mapping
Real time data streaming	N	N	Y	

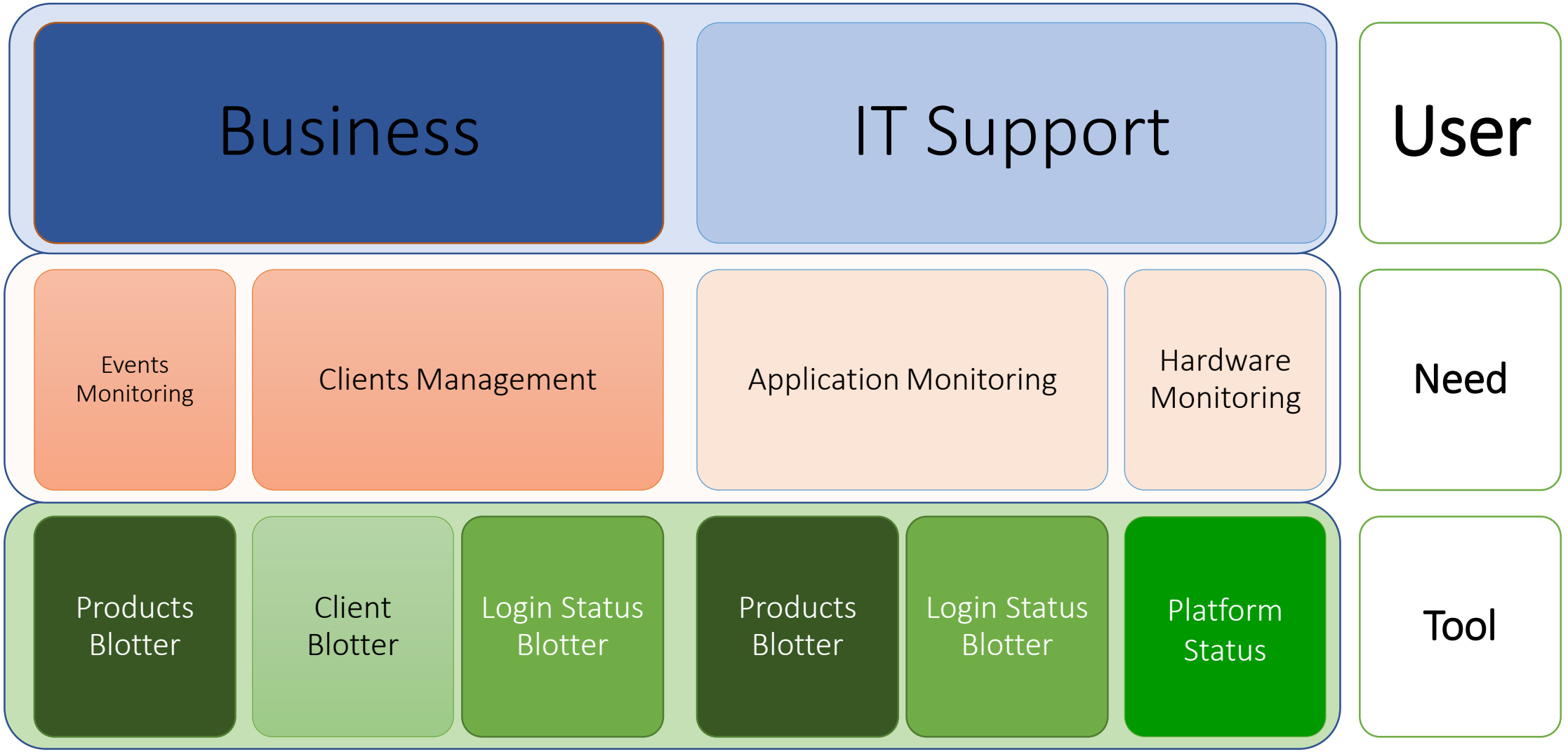
Other possible tools

Tools similar to SpinBoard.

Functionality	Qlik Sense	Tableau	PowerBI	SpinBoard	Notes
gRPC Connector	N	N	Y	Y	• *PowerBI* requires to buy a third party connector
Kafka Connector	N	N	Y	Y	
Aeron Connector	N	N	N	Y	• *PowerBI* data can be pushed via rest api • *Tableau* polls data instead of subscribing
ZMQ Connector	N	N	N	Y	
Real time data streaming	Y	N	N	Y	
Oracle/sql Connector	Y	Y	Y	Y	
Custom Data Segregation	N	Y	Y	Y	
Hierarchical Grids	Y	Y	Y	Y	
Grids filtering / sorting	Y	Y	Y	Y	
Cloud Ready	Y	Y	Y	Y	

A layered tool

One tool, different audience



SpinBoard Use Cases

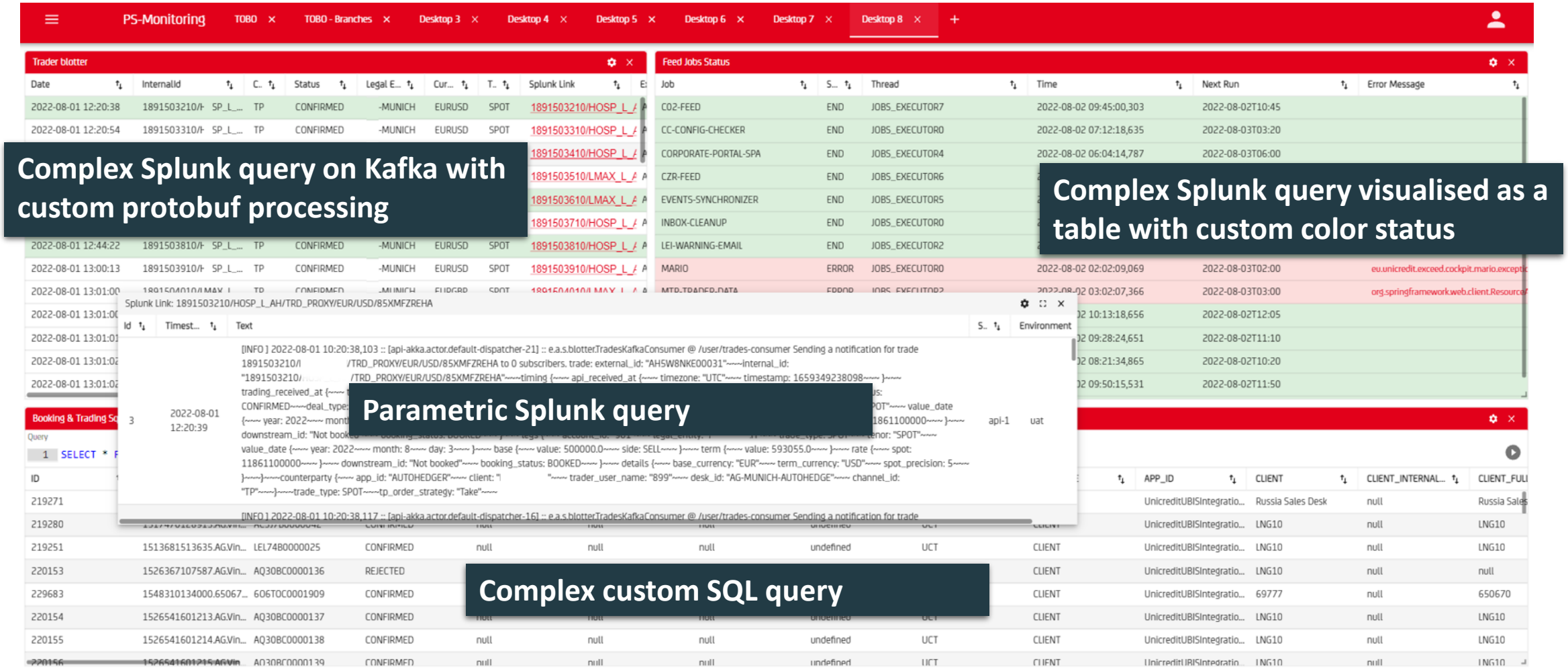
SpinBoard use case #1 : daily orders monitoring

Monitoring Orders status, intercept errors and investigate the cause:

- Before ***SpinBoard***, PS Team had to:
  - Configure an alert on **Splunk**
  - Open **SQL Developer**, write the very specific query for extracting order data
  - On **Splunk**, rewrite the query for investigating the specific cause

- With ***SpinBoard***, PS Team can now:
  - Immediately see in blotters any problematic order. All data is already presented there, click on a cell to open a dialog containing a Splunk query ready to be checked

What does integration really mean?



What does integration really mean?

Integration benefits

- Centralize different applications in a single point:
  - Instead of opening Splunk, Geneos, SQL Developer .. all needed functionalities are grouped together in a single FE

- Transparently integrate heterogeneous information sources:
  - No matter from where information comes from: it is always represented in the same way
  - All grids have the same controls, filtering, sorting and grouping always work in the same way

Integration benefits

Why Integrating

- **Simplified decision making**
  Working on a unified view simplifies the decision making process, removing the need to move between different applications to access data

- **Enhanced analysis**
  Related data is more meaningful & powerful when it is pulled together in one application. Analysis of multiple data sources is better handled by bringing the data together where trends and conclusions can be drawn much sooner

SpinBoard Use Cases

SpinBoard use case #2 : daily checks / post release checks

Monitoring platform status and each service functionalities :

- Before ***SpinBoard***, PS Team had to:
  - Check in **Geneos** each service status
  - Check on **Splunk** specific dashboards
  - Open **UCT** login page and manually verify
  - Open, one by one, **services** REST endpoints to analyse returned data

- With ***SpinBoard***, PS Team can now:
  - Just open the "System Status" tools and check

SpinBoard Use Cases

SpinBoard use case #2 : daily checks / post release checks

SpinBoard Use Cases

SpinBoard use case #3 : commodity deals blotter

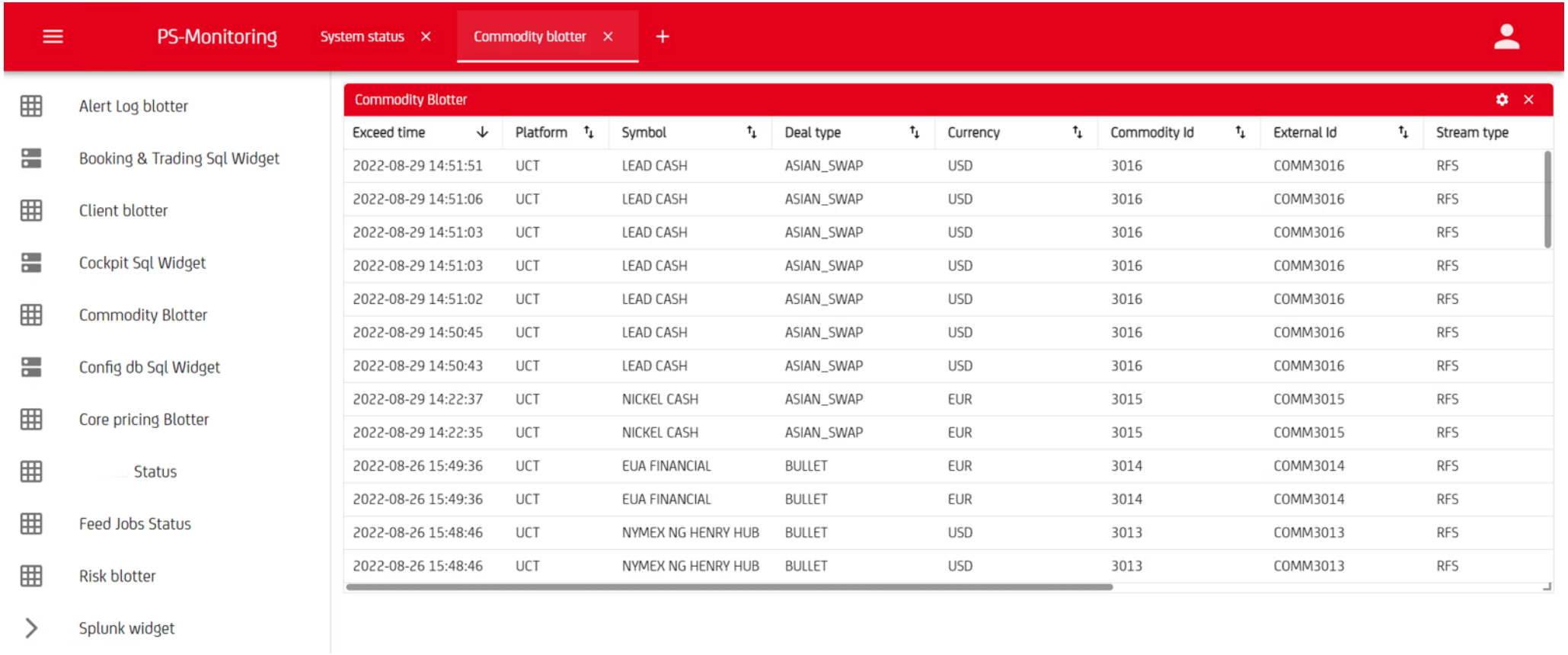
Monitoring the deals on commodities:

- Before ***SpinBoard***, Sales:
  - Does not have any possibility to see these data on any tool.
  - The only solution is to implement and integrate a new blotter using other vendor-provided front-ends

- With ***SpinBoard***, Sales:
  - Can just look at the commodity blotter containing all the information related to commodity deals.
  - This blotter filter out the information based on the Legal entity of the sale (which is necessary for regulatory needs)

SpinBoard Use Cases

SpinBoard use case #3 : commodity deals blotter



SpinBoard Use Cases

SpinBoard use case #4 : daily business trades monitoring

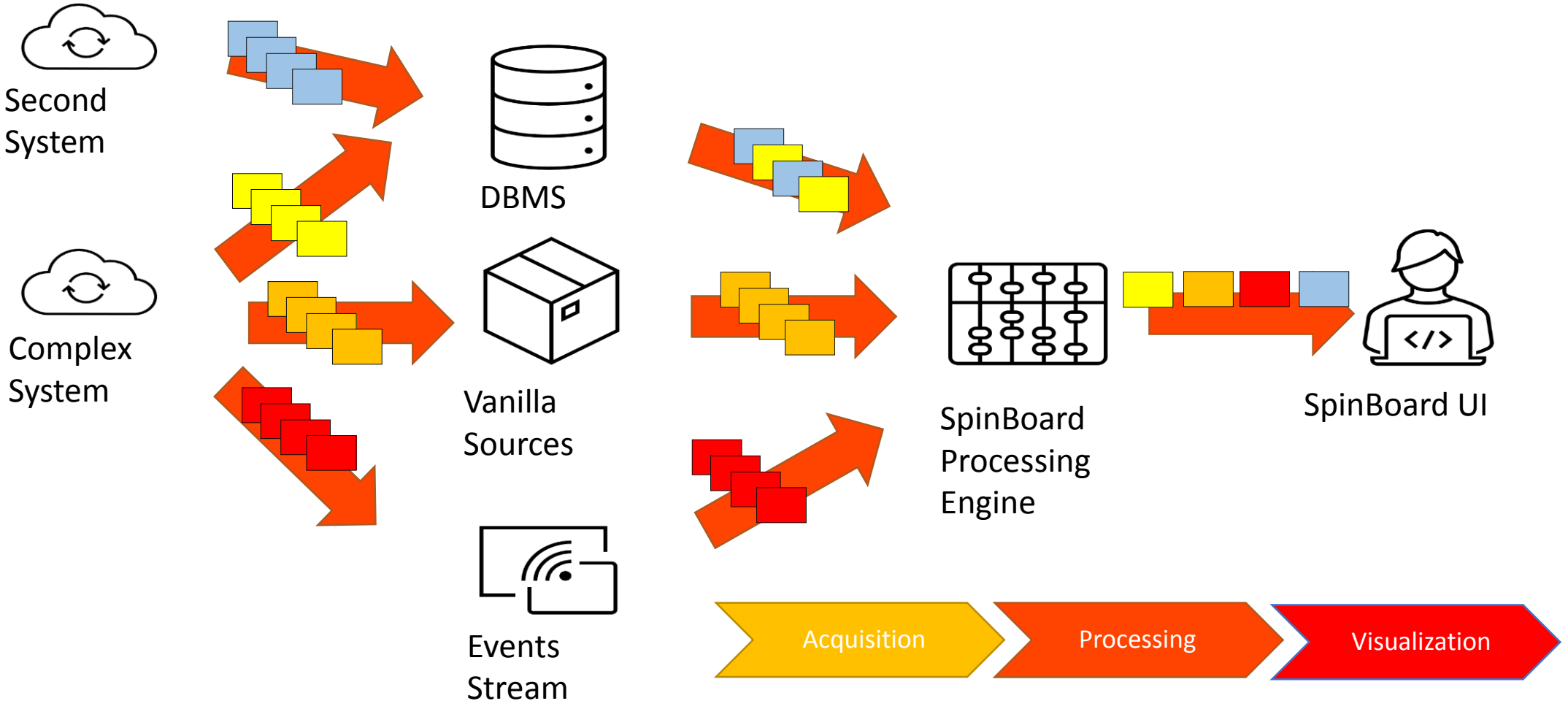
Monitoring platform hedging trades:

- Before ***SpinBoard***, traders had:
  - A flat UCT blotter, hard to inspect and missing some information that can be integrated using other vendor-provided front-ends

- With ***SpinBoard***, Traders will be able to:
  - Just look at a multi-source blotter aggregating trades from kafka, trade-capture deals stored in the DB and tradepoint-client messages from a different kafka topic. All data represented in a single multi-hierarchical blotter

SpinBoard Architecture

A general overview of the SpinBoard architecture



SpinBoard Architecture

Backend structure

Backend can be split into five different sub-components:

1. **Source** fetches the data from an external component
2. **Adapter** converts the data to an internal representation
3. **Core** stores and merges the data from different sources
4. **DataFilter** filters data for the user view
5. **Target** sends the data to the front end

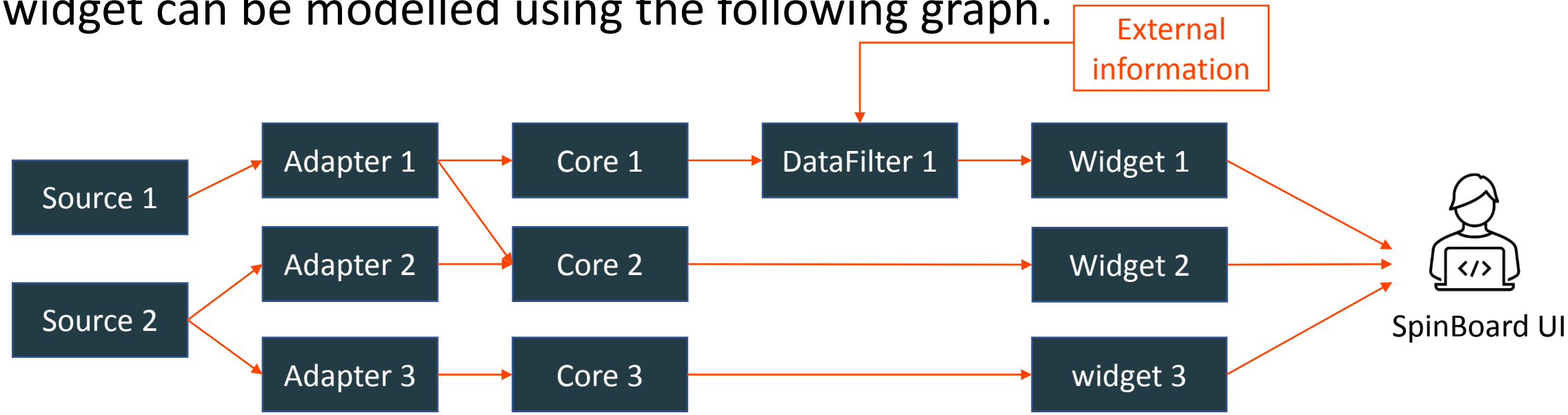
Sub-components just need to be initialized, clients can write their own specific sub-components extending proper interfaces.

FE widgets are the result of a chain or a composition of chains built using above sub-components.

SpinBoard Architecture

The chain structure of the backend

A general setup containing two different sources and three different widget can be modelled using the following graph.



- Widget 1 gets the data from Source 1 through Adapter 1 with a data filter layer.
- Widget 2 contains data from Sources 1 and 2 through respectively Adapters 1 and 2
- Widget 3 gets the data from Source 2

SpinBoard Architecture

The adapter as a stateless mapping component

The goal of an adapter is to map the data from an external source to an internal representation.

The tool provides a general Interface receiving a generic EventMessage containing the data from a source and returning a list of Rows.

```
public interface Adapter {
    List<Row> baseAdapt(EventMessage<?> message);
}
```

Every customer can implement a set of adapters based on the widget requirements:

```
private List<Row> buildRow(EventMessage<KafkaMessage> message) {
    Map<Long, Cell> cells = new HashMap<Long, Cell>();

    cells.put(NAME, CellBuilder.builder().withDisplayValue(message.getName()).build());
    cells.put(ID, CellBuilder.builder().withDisplayValue(message.getId()).build());
    cells.put(PRICE, CellBuilder.builder().withDisplayValue(message.getPrice()).build());

    Row finalRow = newRowBuilder().withId(message.getId()).withAction(INSERT).withCells(cells).build();

    return Lists.newArrayList(finalRow);
}
```

For every KafkaMessage a row containing three cells is created

Then the resulting row is returned to the tool for the next step of the process

Thank You!

Contact us at:

Info@t-spin.com

Visit us at:

https://www.t-spin.com