Using Splunk Data Stream Processor For Advanced Stream Management

David Cornette
Senior MTS | T-Mobile

splunk> .conf19

.conf19

splunk>

David Cornette

Senior MTS, T-Mobile

Michael Guenther

Senior Advisory Engineer, Splunk

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Splunk @ T-Mobile

Handling the ever changing data of a telecommunications company.

.conf19

splunk>

T-Mobile's Daily Ingest

Some numbers that keep us up at night…

Ingested Data Per Day

Sourcetypes (and counting)

Splunk Forwarders

Ingested Events Per Day

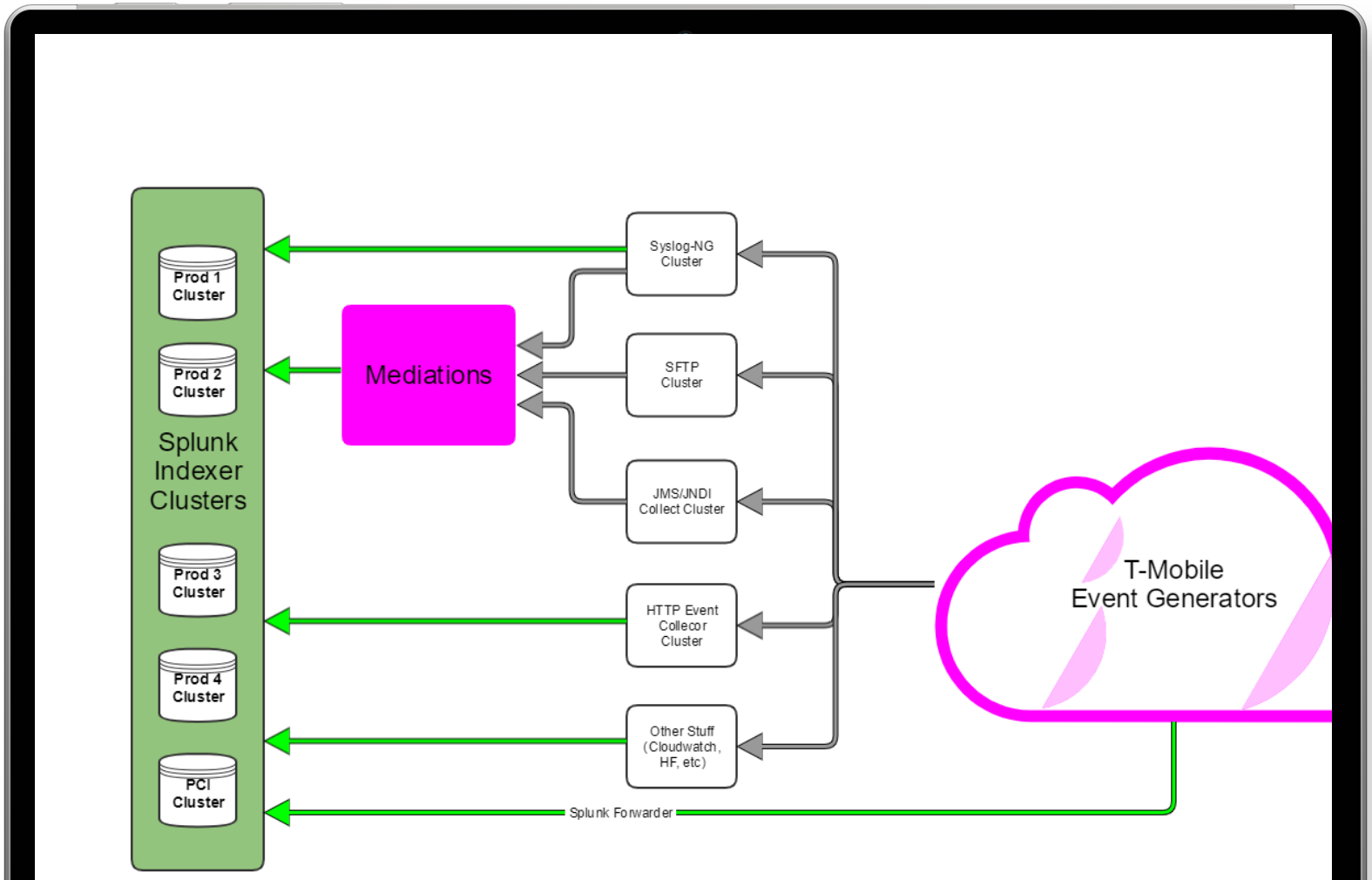
Indexer Row Scans Per Day

(2.5m searches)

Ingest Map

110 billion events/day

(Hint: Not as much forwarder as you think!)



- SFTP: 20 billion/day
  - 100% mediated
- Syslog: 10 billion/day
  - ~20% mediated
- JMS/JNDI: 1 billion/day
  - 100% mediated
- HEC: 15 billion/day
  - Enforced JSON
- UF/HF: ~60 billion/day

Challenges

Dealing with our data

- **Constant Data Drift**
  - Every log generator logs differently
  - Logging is organic & ever-changing
  - Devs are Evil
    - just kidding
      - (sorta)
  - Layers of log collection
  - Data corruption
- **User Perceptions**
  - When logs change, data breaks
  - Best case: “Splunk broken” tickets
  - Worst case: User presumes it’s just the way it is
    - Starts running rex on _raw to search
- **So Much Data!!**
  - Splunk historically cannot pre-process at these volumes
  - Splunk’s “Service Edge” pulls inside the processing layer
  - Dozens of Perl & Python scripts required to handle mediations
  - Huge & complex records inflating ingest volumes

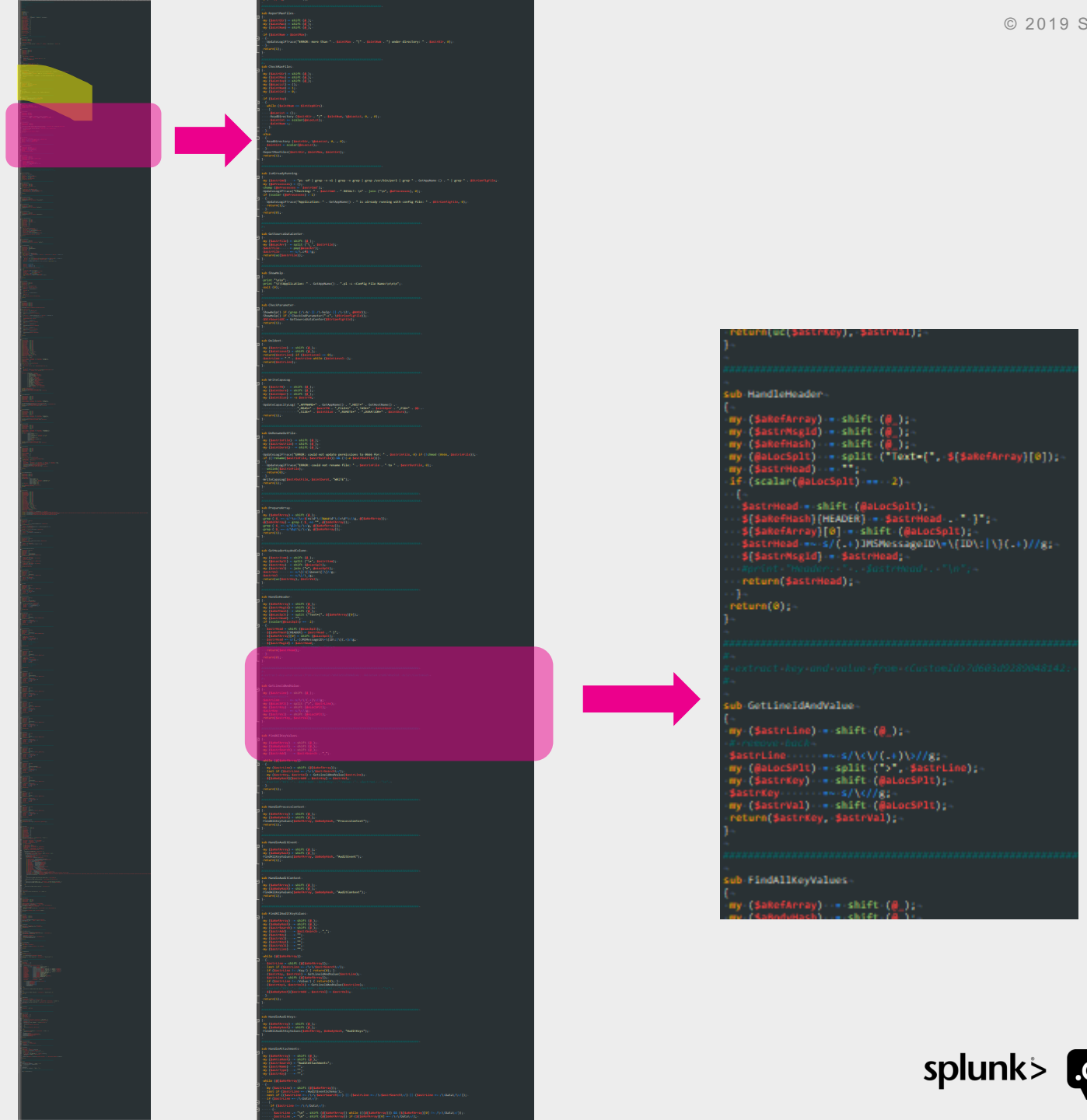
The Mediations Layer

(and Why it's Doomed)

- Complexity
  - Dozens of Perl & Python scripts across dozens of hosts in several clusters
  - Growth & Scaling is per-stream, uncontrolled & difficult
  - Not multi-threading
  - Disk-intensive
    - Disk write on input, disk write on output
    - Splunk forwarders (several per process) required to move data fast enough to indexers
- No manageability
  - Code-level changes as needed when logs change
- No visibility
  - We have to watch data rates at the indexer or mediations layer logging to watch rates and catch failures
- Routing?
  - Super, as long as it's to an indexer…
- To Be Fair!
  - We push something around 25 billion events per day through mediations on the way to Splunk.

Sometimes you just have to see it…

A Mediations Script Example



What We Need

Make Data Processing Our Super Power

- Simplicity
  - Single platform with many options
- Scalability
  - Stop the single-threaded madness!!!
  - Refer to 25-billion-event-per-day current load
    - Now triple it just for starters
- Route Flexibility
  - Better Input and Output flexibility
  - Not just a rigid pass-through processor
  - Consumer & Producer, many sources, many destinations
  - Kafka integration
- Manageability
  - Modular and re-usable structures instead of monolithic scripts
- Visibility
  - How are my pipelines performing?
  - Is something impeding data flow?
- Data Flexibility
  - More than just event breaking
  - Enrich, Filter, Aggregate – all at once!
- … and I want it inside Splunk's "Service Edge"

Solution:

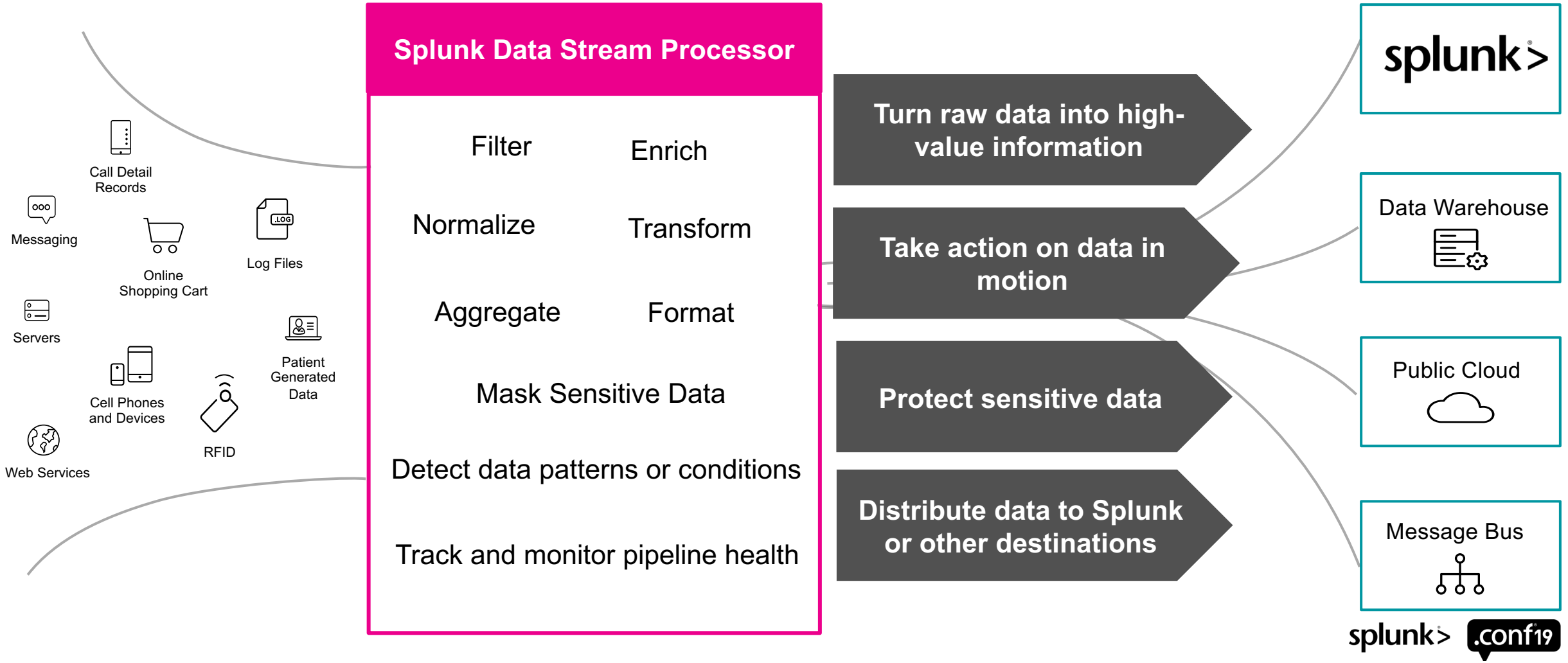
Data Stream Processor

.conf19

splunk>

Splunk Data Stream Processor

A real-time stream processing solution that collects, processes and delivers data to Splunk and other destinations in milliseconds



splunk> .conf19

DSP Capabilities and Requirements

✓ **Supported Data Sources*:** Kafka, Kinesis, S3, CloudTrail, Event Hubs, REST APIs, Splunk (Universal Forwarder, Heavy Weight Forwarder)

✓ **Supported Destinations*:** Kafka, Kinesis, Splunk

✓ **Infrastructure Based Pricing (vCPUs)**

Hardware Requirements

- Minimum Node Requirement
  - CPU: 8 core (16 recommended)
  - Memory: 64GB (128GB recommended)
  - Network: 10Gbps
  - Storage: 1TB
- Minimum 5 Node Cluster

Deconstructing Complex Data

.conf19
splunk>

Retail and Application Data

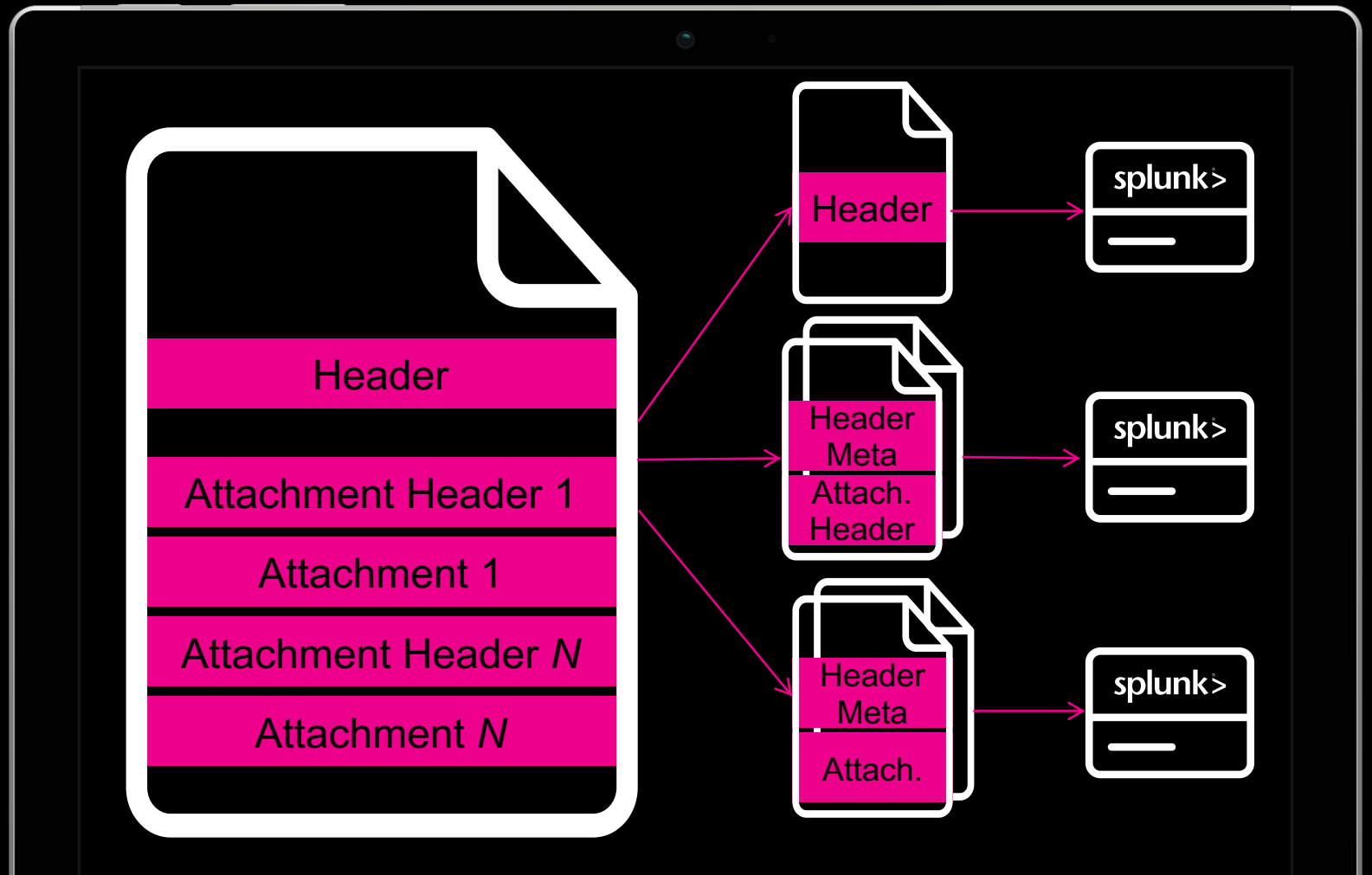
Rebellion Data Format

- Imposing Order
  - Both external and internal data providers use this format
  - Enforces conformity among all sources at the cost of complexity
- Goal
  - Parse and Break Events in-flight keeping key fields
  - Route to pieces to different indices

Demo

Rebellion Pipeline

Simplified Streaming Process



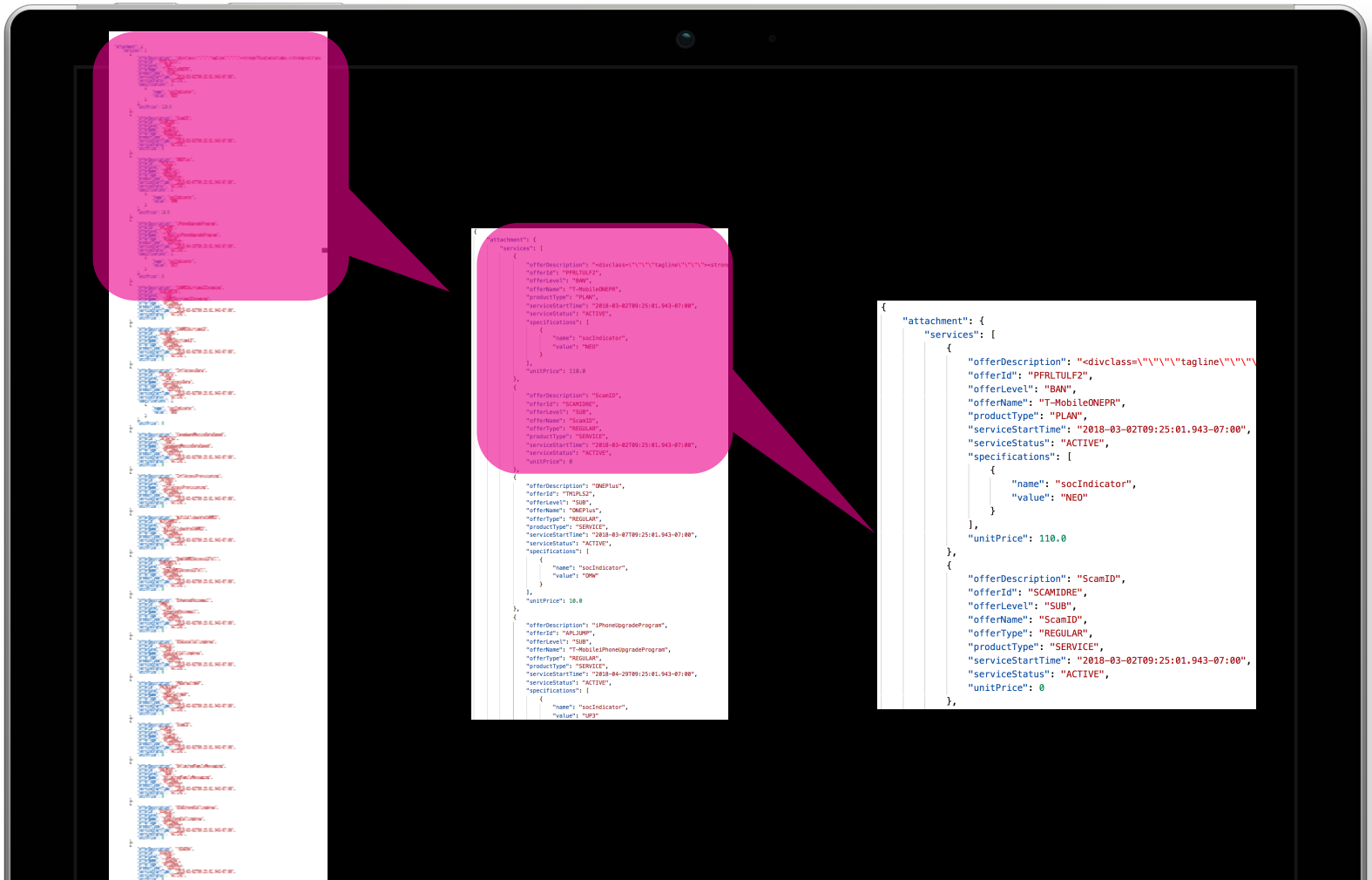
- Processing Events
  - Envelopes and Attachments separated
  - Header metadata is added to each new event
  - Events sent to the relevant indices
- Malformed Data Handling
  - Malformed events sent to a DLQ in Kafka

Filtering The Noise

.conf19
splunk>

Offers JSON Event

Reducing the Noise



- Only a fraction of the data has value
  - Users are only interested in non-zero value offer data
- Goal
  - Break offers out into individual events
  - New events contain original header metadata
  - Filter out zero value offers

Demo

Before DSP

- Two Scenarios
  - Pre-processing scripts on forwarders
  - Potentially complex SPL requiring an admin

After DSP

- Centralized Processing
  - No pre-processing scripts on forwarders
  - JSON converted to KV
- Simplified SPL
- Noise Reduction
  - Zero values are aggregated into counts for later reference
  - Reduced indexer load

DSP Pipeline

Previously

After

Offer With Value

10/11/19 10:55:56.000 AM

offerTimestamp=Oct 11 2019 15:55:56 eventId=2c14d96b-51a5-4102-8d8e-3767c80f8af0 offerId=USAEHA offerName=^USAEHA serviceStartTime=Oct 11 2019 15:55:56 productType=SERVICE serviceStatus=ACTIVE offerLevel=SUB unitPrice=37.0 info=1540484701953 offerDescription=^USAEHA

host = 127.0.0.1 | source = eventgen | sourcetype = tmo:offers

Aggregated Count of Zero Value Offers

10/11/19 11:00:20.000 AM

eventId=1a5e0179-8ea6-48d3-8cf8-3162dd340420 offerType=OPTIONAL offerCount=9 windowStart=1570809610000 windowEnd=1570809620000 windowTrigger=1570809619999

host = dsp | source = dsp:offers:zeroes | sourcetype = tmo:offers:zeroes

splunk> .conf19

In Summary

DSP gives us the tools to reimagine our ingest pipeline.

1. Centralizing and simplifying the management of complex data structures and pre-processing tasks
2. Lowering barriers to a modern streaming data solution in a Splunk-native environment
3. Real-time actionability during the ingest process
4. Sharing data at scale and in real time with partner solutions

Q&A

.conf19
splunk>

Other Data Stream Processor Sessions

1. DEV1317 - Data Stream Processor: Architecture and SDKs
2. FN1987 - Using Splunk Data Stream Processor as a streaming engine for Apache Kafka
3. DEV1139 – Detecting Anomalies in DSP Pipelines Using Real Time Machine Learning

.conf19

splunk>

Thank You!