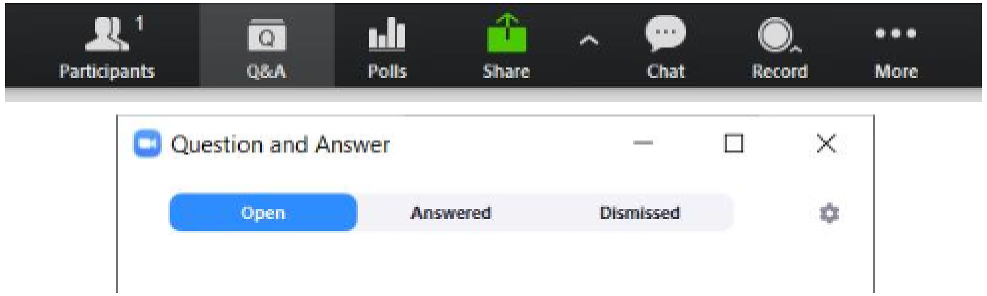
An Introduction to Data Engineering Streaming (*AKA* Big Data Streaming)

Ramesh Jha

Informatica Global Customer Support

Informatica™

Housekeeping Tips



- Todays Webinar is scheduled to last 1 hour including Q&A
- All dial-in participants will be muted to enable the speakers to present without interruption
- Questions can be submitted to “All Panelists" via the Q&A option and we will respond at the end of the presentation
- The webinar is being recorded and will be available to view on our INFASupport YouTube channel and Success Portal. The link will be emailed as well.
- Please take time to complete the post-webinar survey and provide your feedback and suggestions for upcoming topics.

Success Portal

https://success.informatica.com

Learn. Adopt. Succeed.

Bootstrap product trial experience

Enriched Onboarding experience

FREE Product Learning Paths and weekly Expert sessions

Informatica Concierge with Chatbot integrations

Tailored training and content recommendations

Safe Harbor

The information being provided today is for informational purposes only. The development, release, and timing of any Informatica product or functionality described today remain at the sole discretion of Informatica and should not be relied upon in making a purchasing decision.

Statements made today are based on currently available information, which is subject to change. Such statements should not be relied upon as a representation, warranty or commitment to deliver specific products or functionality in the future.

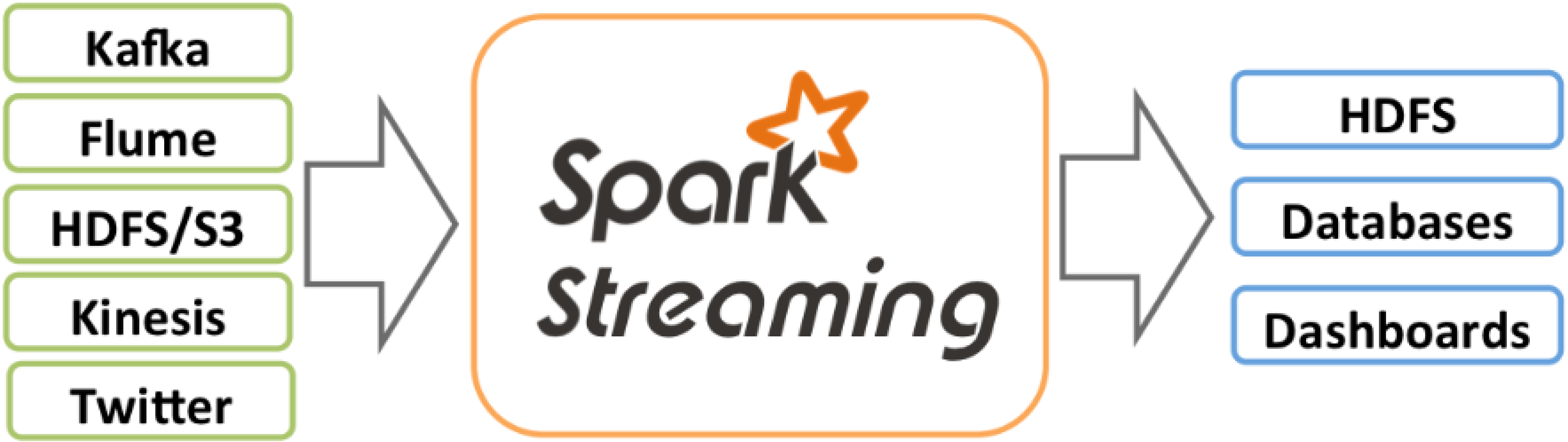
Agenda

- Streaming Overview
- Structured streaming
- Streaming Sources and Targets
- Streaming mapping Configurations
- Window transformation
- Use case & Demo
- Troubleshooting and self-service
- References
- Q&A

Streaming Overview

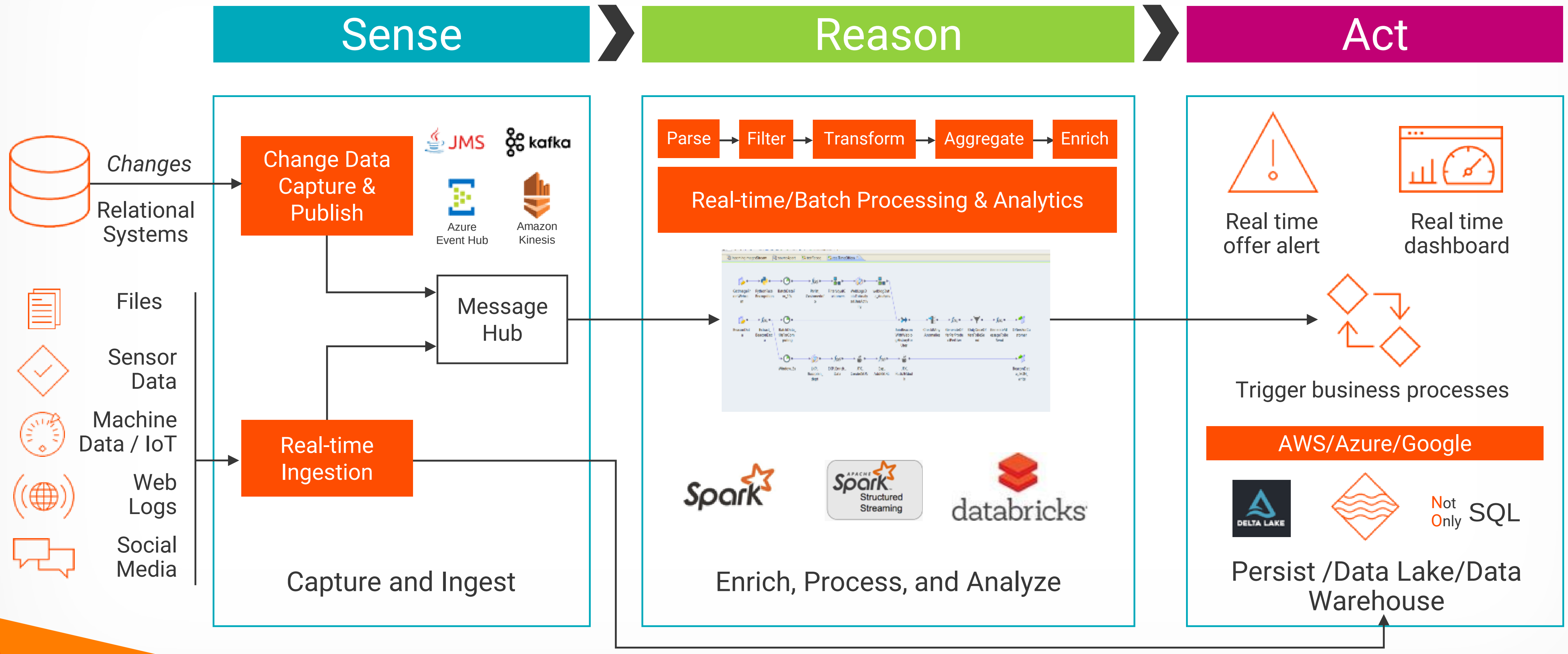
Streaming Overview

Streaming is the processing of live data streams from unbounded data sources like Kafka, Flume, Kinesis, TCP sockets.



An unbounded data source is one where data is continuously flowing in and there is no definite boundary

Streaming Overview – Informatica Data Engineering Streaming



Sense

Reason

Act

Changes

Relational Systems

Files

Sensor Data

Machine Data / IoT

Web Logs

Social Media

Change Data Capture & Publish

JMS

kafka

Azure Event Hub

Amazon Kinesis

Message Hub

Real-time Ingestion

Capture and Ingest

Parse → Filter → Transform → Aggregate → Enrich

Real-time/Batch Processing & Analytics

Spark

Apache Spark Structured Streaming

databricks

Enrich, Process, and Analyze

Real time offer alert

Real time dashboard

Trigger business processes

AWS/Azure/Google

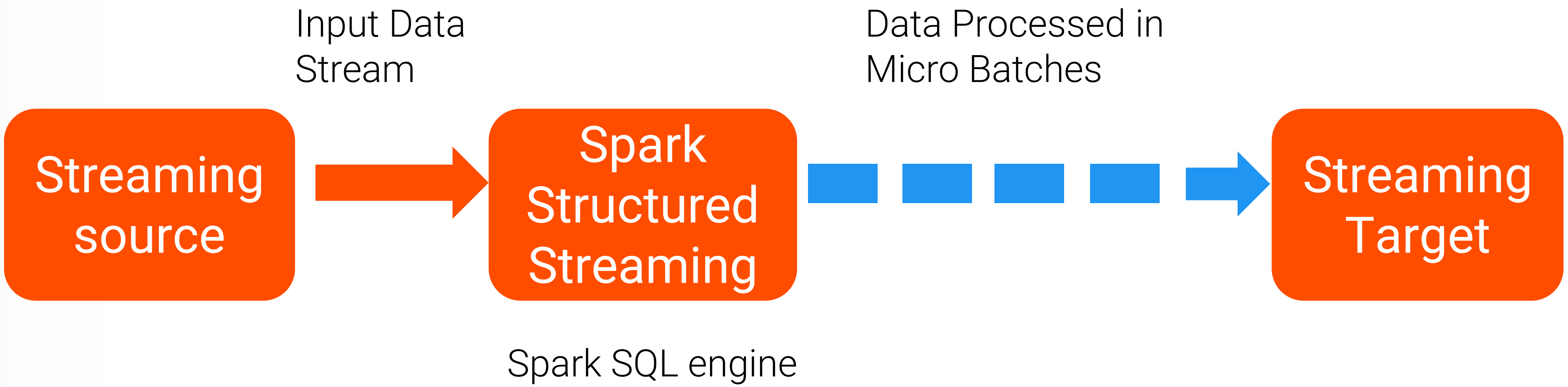
DELTA LAKE

Not Only SQL

Persist /Data Lake/Data Warehouse

Informatica

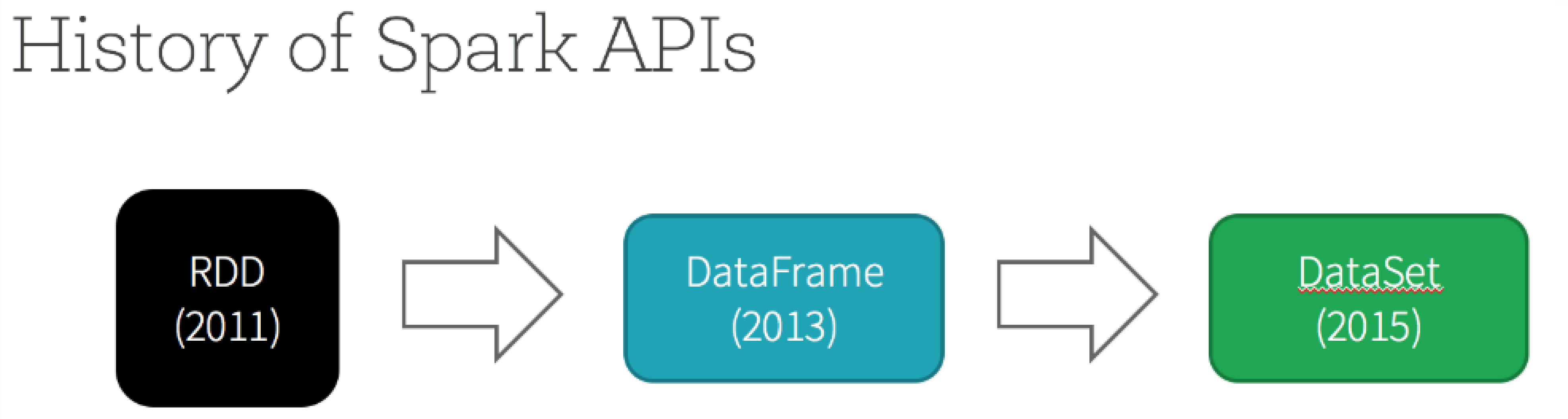
Streaming Process



Spark Structured Streaming receives data from streaming sources such as Kafka and divides the data into micro batches.

Structured Streaming

Structured Streaming



History of Spark APIs

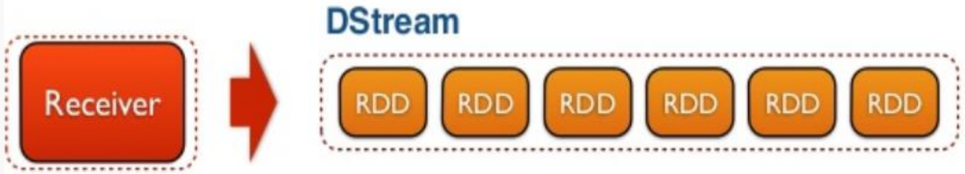
- Spark Core API
- SparkContext
- Low Level API

- Spark SQL
- RDD + Schema
- SqlContext
- Optimizer support
- High Level(Built on RDD)

- Extension of DataFrame.
- Type Safety

Structured Streaming

Spark Streaming
(Pre 10.2.2)



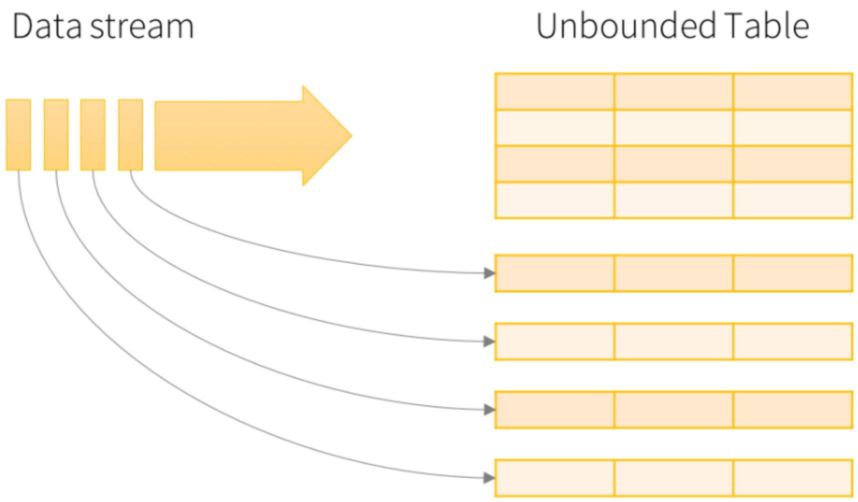
e.g: Kafka,
Kinesis,
Flume,
Sockets,
Akka
etc

Data **Collected**, **Buffered** and **Replicated** by a **Receiver** (one per DStream) then **Pushed** to a **stream** as small **RDDs**

Configurable **Batch Intervals.**
e.g: 1 second, 5 seconds, 5 minutes

RDD

Structured Streaming
(10.2.2 & beyond)



new data in the
data stream
=
new rows appended
to a unbounded table

Data stream as an unbounded table

DataFrame

Structured Streaming – Why ?

Leverage Spark Optimization

Dstream cannot leverage the optimizations offered by Spark SQL's Catalyst optimizer and Spark's Tungsten Optimization especially managing Aggregator state management.

DataFrame - Can leverage all the optimizations offered by Spark SQL Catalyst optimizer and Tungsten Optimization.

Handling Late Data

This is exclusively a Structured Streaming , we can control how late the window can wait before it can evicted from Result Table and written to target through Watermark property

Output mode

There is no output mode in Dstream. It is append by default.
Determine how and when data needs to be evicted from Result Table to the target. Supported output modes are Append , Update and Complete

Message delivery

Guarantees exactly once message delivery

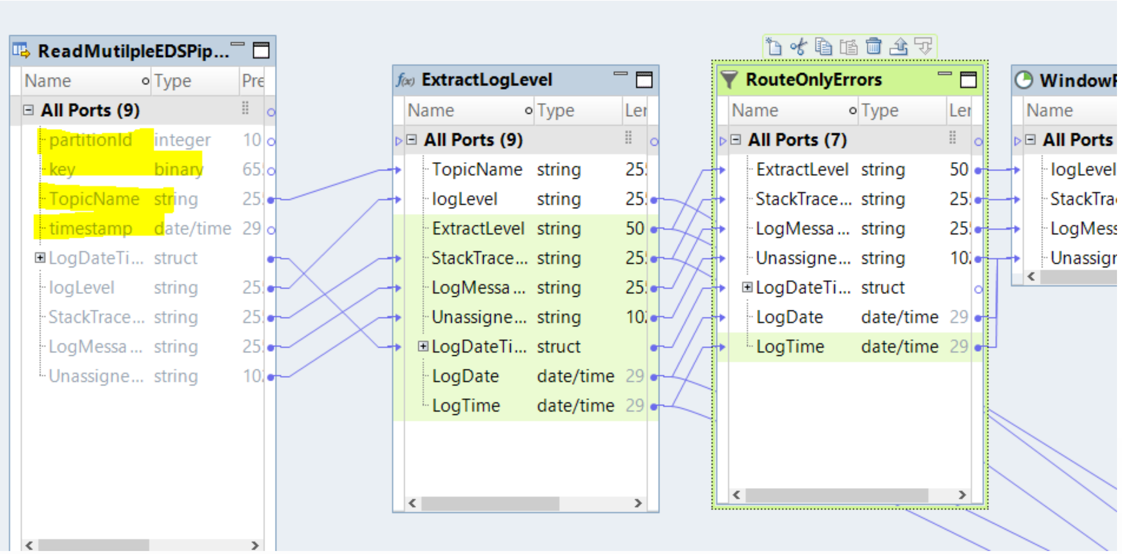
Structured Streaming – Why ? Contd..

Message Header Support

- Enable developers to use message headers from streaming sources
- Transformations can be applied on message header data

How does It help ?

- Customers can now use message metadata for better analytics on the data.
- No need to parse the whole message.



ReadMutilpleEDSPip...

Name	Type	Pre
⊟ All Ports (9)		
partitionId	integer	10
key	binary	65
TopicName	string	25
timestamp	date/time	29
⊞ LogDateTi...	struct	
logLevel	string	25
StackTrace...	string	25
LogMessa ...	string	25
Unassigne...	string	10

ExtractLogLevel

Name	Type	Ler
⊟ All Ports (9)		
TopicName	string	25
logLevel	string	25
ExtractLevel	string	50
StackTrace...	string	25
LogMessa ...	string	25
Unassigne...	string	10
⊞ LogDateTi...	struct	
LogDate	date/time	29
LogTime	date/time	29

RouteOnlyErrors

Name	Type	Ler
⊟ All Ports (7)		
ExtractLevel	string	50
StackTrace...	string	25
LogMessa ...	string	25
Unassigne...	string	10
⊞ LogDateTi...	struct	
LogDate	date/time	29
LogTime	date/time	29

WindowF

Name
⊟ All Ports
logLevel
StackTra
LogMess
Unassigr

Streaming Sources and Targets

Streaming Sources and Targets

Spark

Sources

- Kafka
- JMS
- Amazon Kinesis
- Azure Event Hubs
- Confluent Kafka
- MapR Streams

Targets

- Kafka
- JMS
- Amazon Kinesis
- Azure Event Hubs
- Confluent Kafka
- HBase
- MapR Streams
- Amazon S3
- Complex file Data Object
- ADLS Gen1,Gen2
- Hive
- JDBC compliant Relational Database.
- Snowflake

Streaming Sources and Targets

DataBricks (Azure)

Sources

- Azure Event Hubs

Targets

- Azure Event Hubs
- ADLS Gen2
- Databricks Delta Lake

Streaming Sources and Targets : File Formats

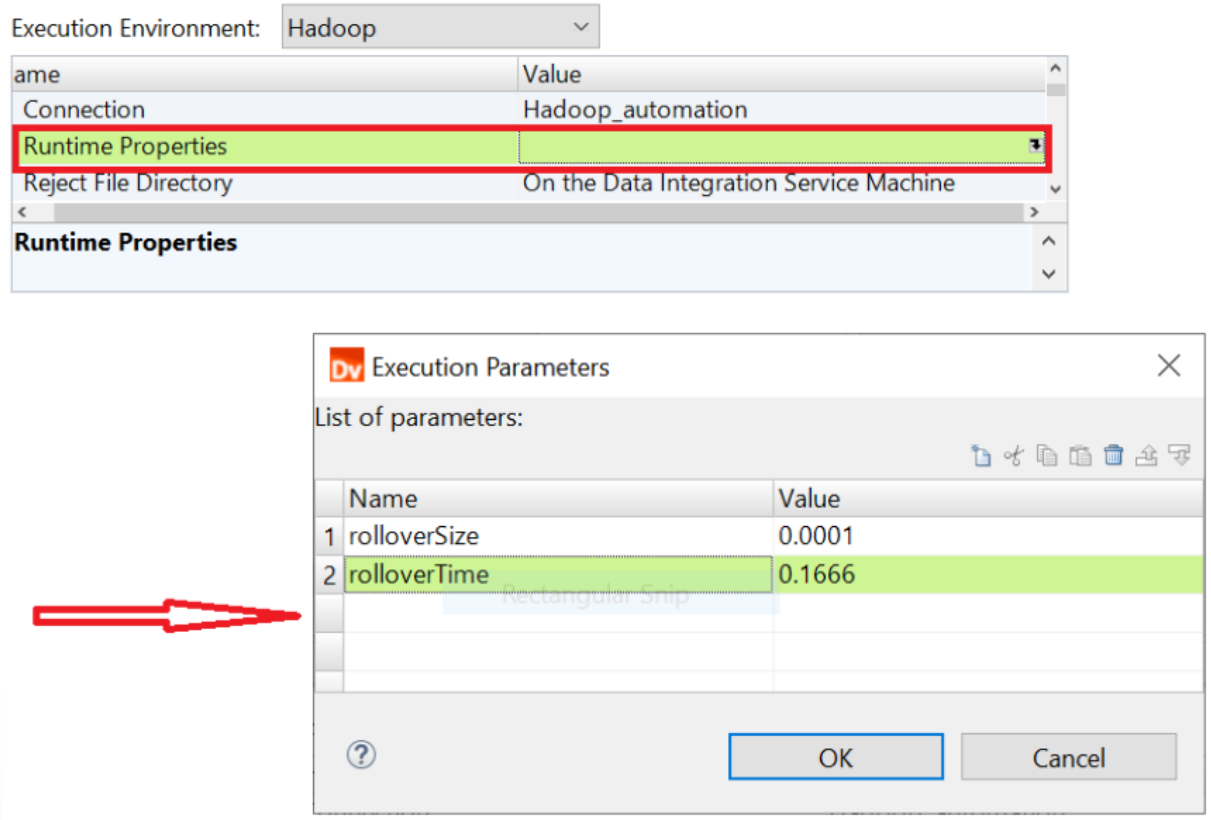
The following table shows the different file formats supported in Data Engineering Streaming :

Format	Schema Type	Amazon Kinesis Firehose	Amazon S3	Azure Data Lake Store	Azure Event Hub	Complex File	JMS	Kafka	MapR Streams
Avro	Flat	Not supported	Supported	Supported	Supported	Supported	Not supported	Supported	Supported
Avro	Hierarchical	Not supported	Supported	Supported	Supported	Supported	Not supported	Supported	Supported
Binary	Binary	Supported	Not Supported	Supported	Supported	Supported	Supported	Supported	Supported
Flat	Flat	Not Supported	Supported	Not Supported	Supported	Not supported	Supported	Supported	Not Supported
JSON	Flat	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported
JSON	Hierarchical	Supported	Supported	Supported	Supported	Supported	Supported	Supported	Supported
XML	Flat	Not supported	Not Supported	Supported	Supported	Supported	Supported	Supported	Supported
XML	Hierarchical	Not supported	Not Supported	Supported	Supported	Supported	Supported	Supported	Supported

Streaming mapping Configurations

Streaming mapping Configurations

- It must have a streaming source.
- For File based Targets, DES provides rollover mechanism of the output file, for downstream application to consume the data seamlessly.
  - Complex file Data object
  - S3
  - ADLS gen1,gen2



Execution Environment: Hadoop

ame	Value
Connection	Hadoop_automation
Runtime Properties	
Reject File Directory	On the Data Integration Service Machine

Runtime Properties

Execution Parameters

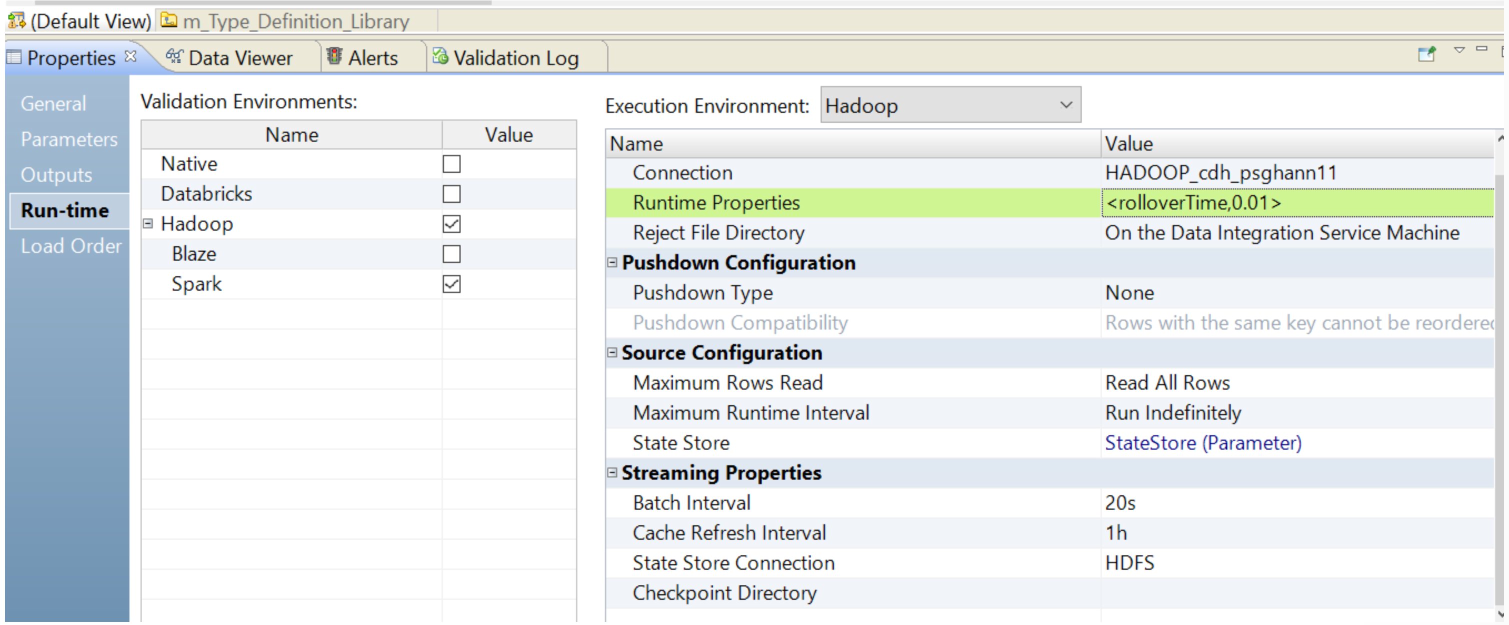
List of parameters:

	Name	Value
1	rolloverSize	0.0001
2	rolloverTime	0.1666

Streaming mapping Configurations

Streaming properties

- Batch interval
- Cache refresh interval
- State Store Connection
- Checkpoint Directory



Validation Environments:

Name	Value
Native	□
Databricks	□
⊟ Hadoop	☑
Blaze	□
Spark	☑

Execution Environment: Hadoop

Name	Value
Connection	HADOOP_cdh_psghann11
Runtime Properties	<rolloverTime,0.01>
Reject File Directory	On the Data Integration Service Machine
⊟ **Pushdown Configuration**	
Pushdown Type	None
Pushdown Compatibility	Rows with the same key cannot be reordere
⊟ **Source Configuration**	
Maximum Rows Read	Read All Rows
Maximum Runtime Interval	Run Indefinitely
State Store	StateStore (Parameter)
⊟ **Streaming Properties**	
Batch Interval	20s
Cache Refresh Interval	1h
State Store Connection	HDFS
Checkpoint Directory	

- Window transformation

Window Transformation

In a streaming mapping, depending on your use case, you might want to apply some aggregation over data collected by time (say, every 5 minutes or every hour), e.g

- Average speed of vehicles every 5 min
- Calculate Maximum value of a stock every min

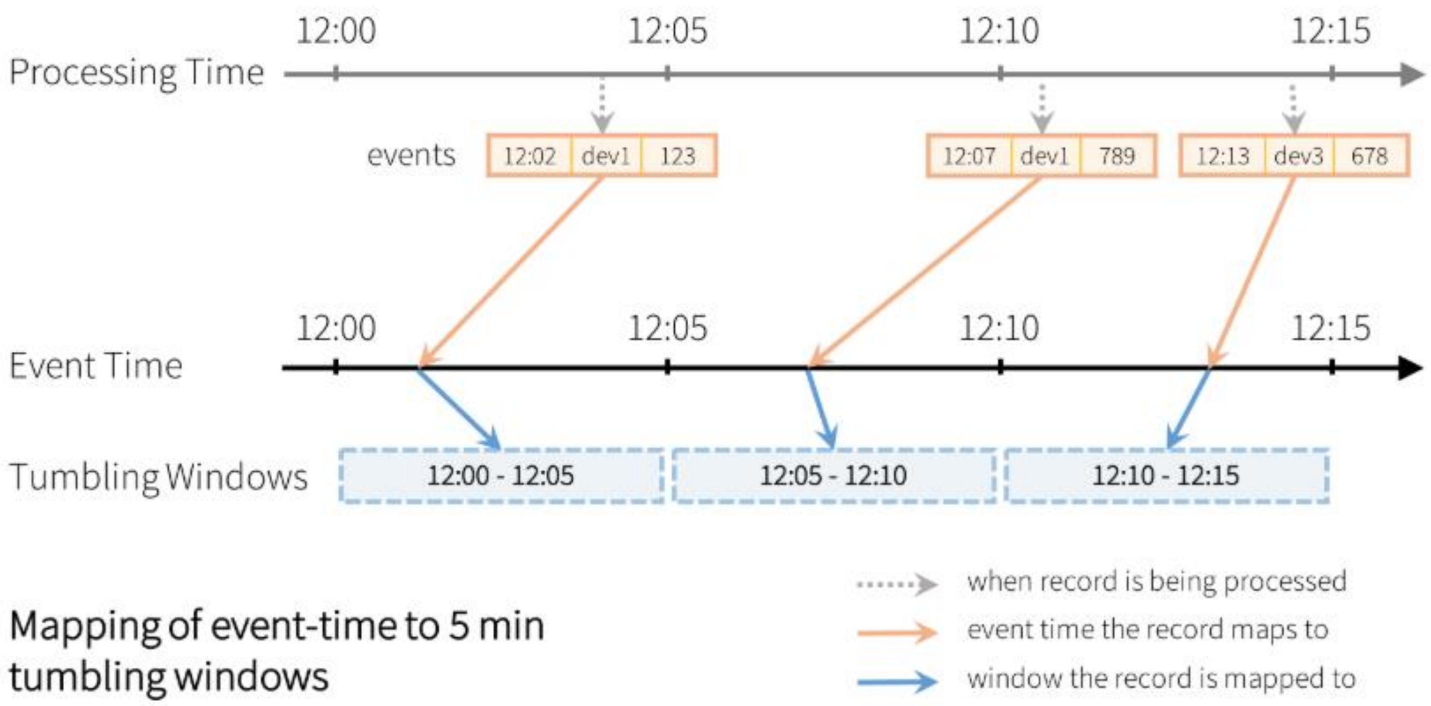
So, To introduce bounded intervals to unbounded data, use a **Window transformation.**

Window Types:

- Tumbling : Max value of a stock price **every five minutes** for stock prices **collected over a five-minute** time interval
- Sliding : Max value of a stock price **every minute** for stock prices **collected over a five-minute** time interval

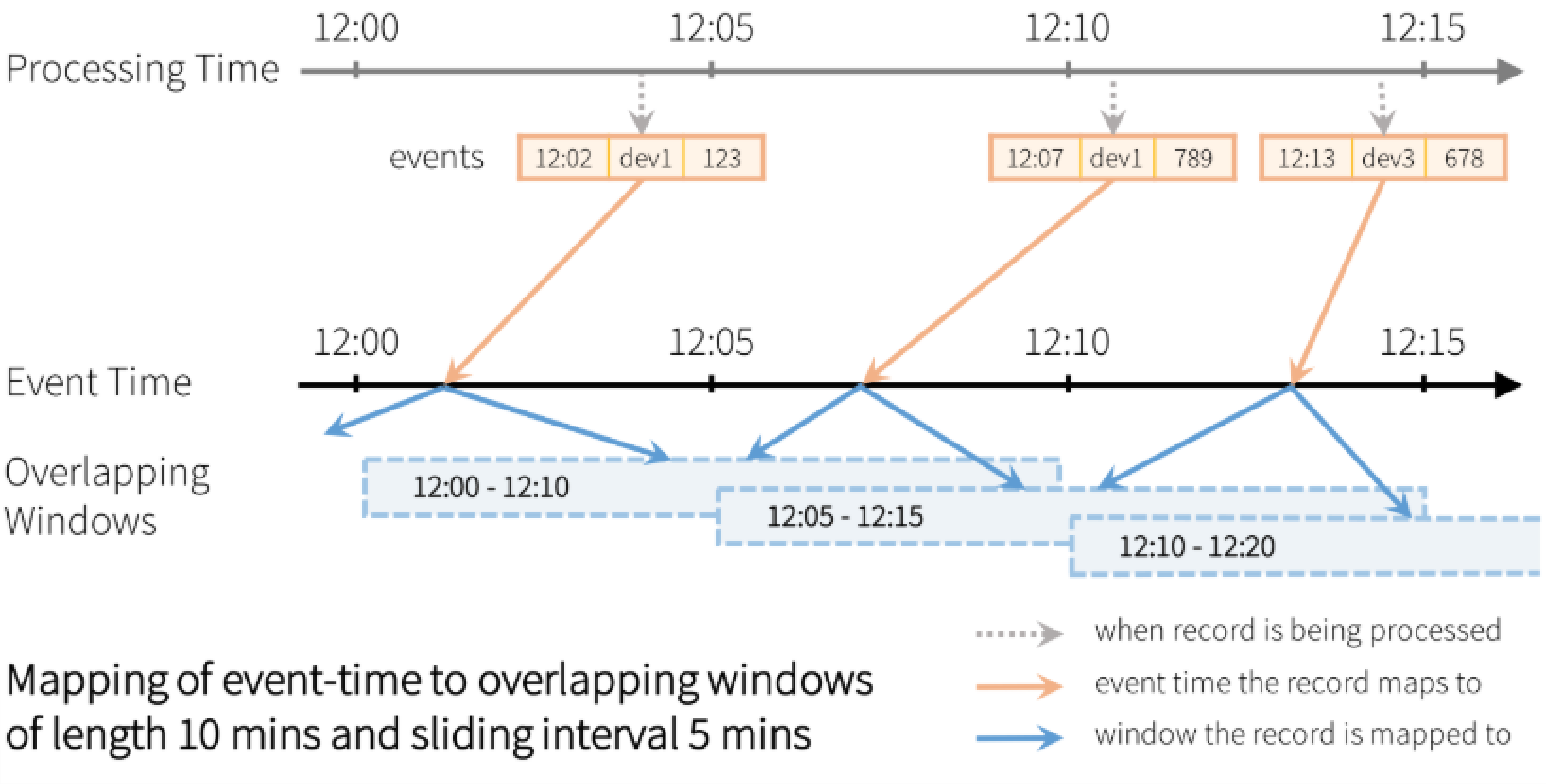
Window Transformation - Tumbling

Every record is going to be assigned to a 5 minute tumbling window as illustrated below



Window Transformation - Sliding

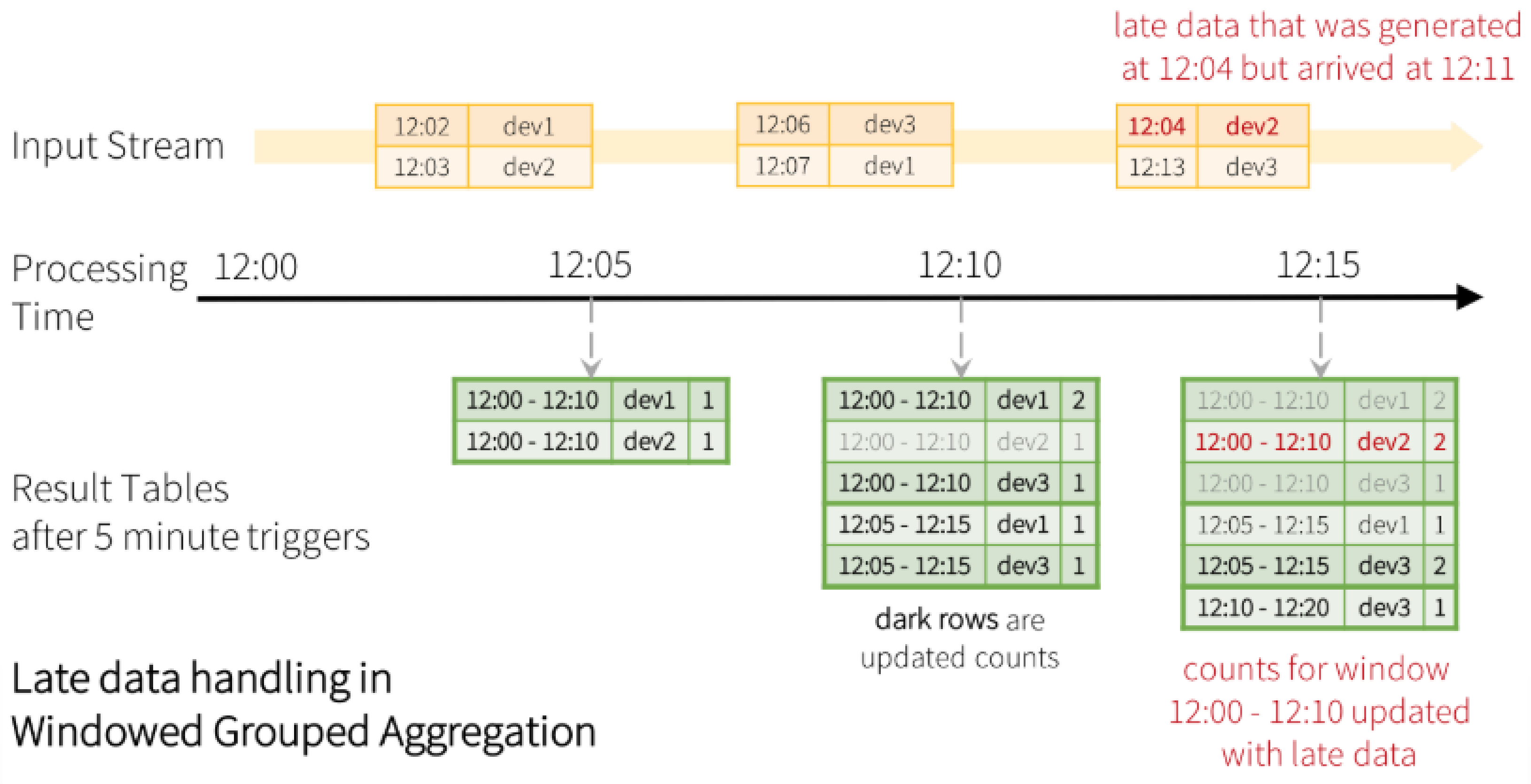
Every record will be assigned to multiple overlapping windows as illustrated below.



Mapping of event-time to overlapping windows of length 10 mins and sliding interval 5 mins

Window Transformation - Sliding

Automatically handles late and out-of-order data



late data that was generated at 12:04 but arrived at 12:11

Input Stream

12:02	dev1
12:03	dev2

12:06	dev3
12:07	dev1

12:04	dev2
12:13	dev3

Result Tables after 5 minute triggers

12:00 - 12:10	dev1	1
12:00 - 12:10	dev2	1

12:00 - 12:10	dev1	2
12:00 - 12:10	dev2	1
12:00 - 12:10	dev3	1
12:05 - 12:15	dev1	1
12:05 - 12:15	dev3	1

dark rows are updated counts

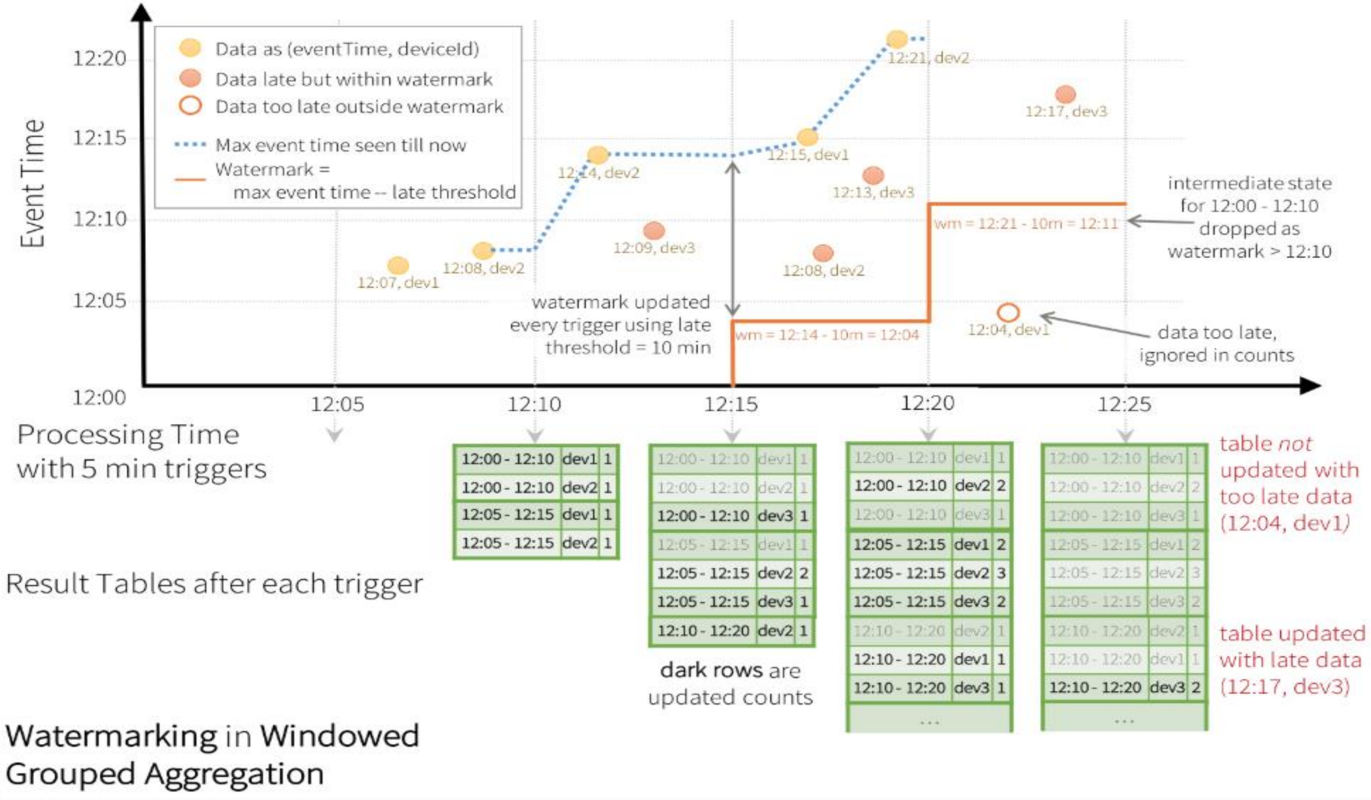
12:00 - 12:10	dev1	2
12:00 - 12:10	dev2	2
12:00 - 12:10	dev3	1
12:05 - 12:15	dev1	1
12:05 - 12:15	dev3	2
12:10 - 12:20	dev3	1

counts for window 12:00 - 12:10 updated with late data

Late data handling in Windowed Grouped Aggregation

Window Transformation - WaterMark

The watermark delay defines threshold time for a delayed event to be accumulated into a data group. ***“Watermark delay”*** gets computed at the beginning of every batch based on the latest data arrived in the previous batch.



12:10

12:00 - 12:10	dev1	1
12:00 - 12:10	dev2	1
12:05 - 12:15	dev1	1
12:05 - 12:15	dev2	1

12:15

12:00 - 12:10	dev1	1
12:00 - 12:10	dev2	1
12:00 - 12:10	dev3	1
12:05 - 12:15	dev1	1
12:05 - 12:15	dev2	2
12:05 - 12:15	dev3	1
12:10 - 12:20	dev2	1

12:20

12:00 - 12:10	dev1	1
12:00 - 12:10	dev2	2
12:00 - 12:10	dev3	1
12:05 - 12:15	dev1	2
12:05 - 12:15	dev2	3
12:05 - 12:15	dev3	2
12:10 - 12:20	dev2	1
12:10 - 12:20	dev1	1
12:10 - 12:20	dev3	1
...		

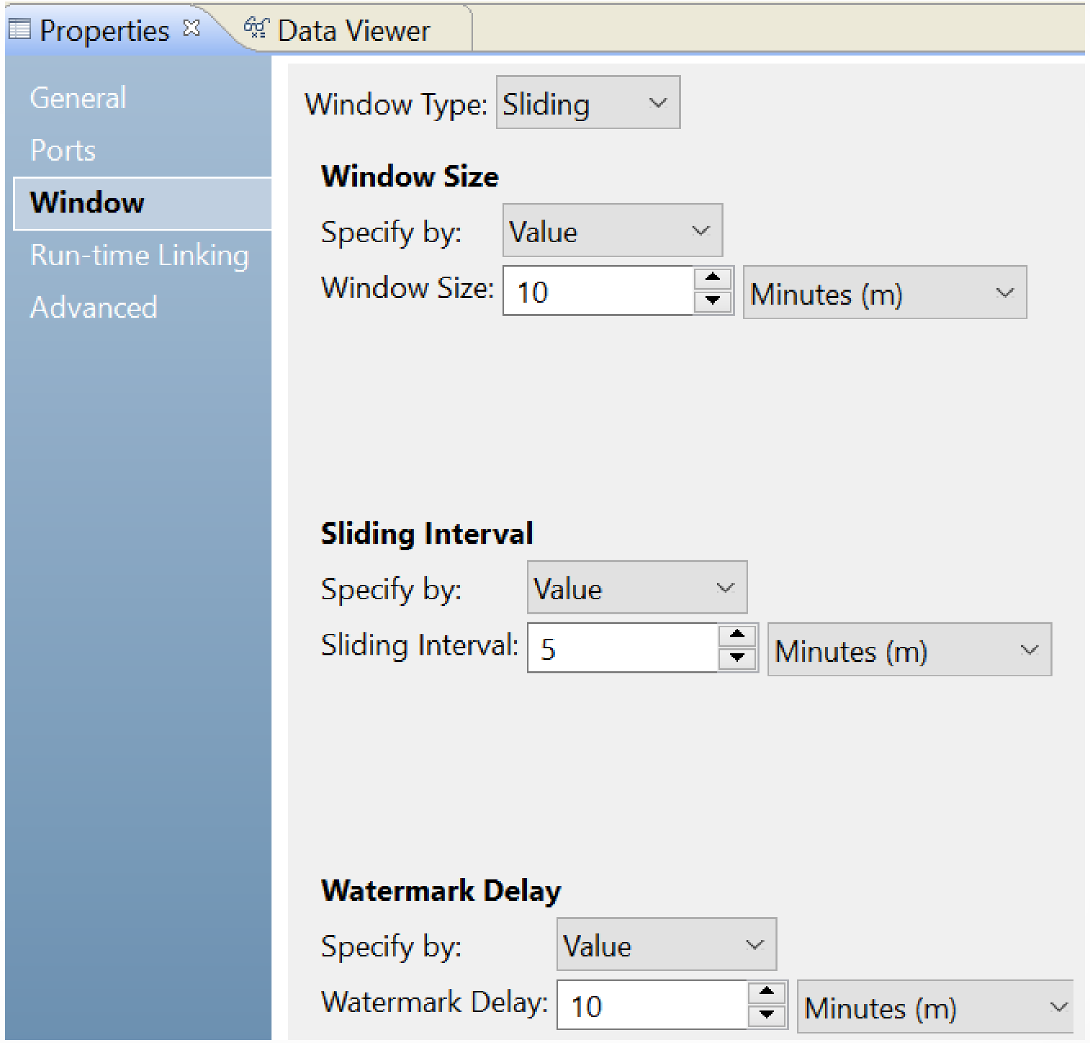
12:25

12:00 - 12:10	dev1	1
12:00 - 12:10	dev2	2
12:00 - 12:10	dev3	1
12:05 - 12:15	dev1	2
12:05 - 12:15	dev2	3
12:05 - 12:15	dev3	2
12:10 - 12:20	dev2	1
12:10 - 12:20	dev1	1
12:10 - 12:20	dev3	2
...		

Watermarking in Windowed Grouped Aggregation

Window Transformation – sum up

- **Window Type**
- **Window Size**
- **Sliding Interval**
- **Watermark Delay**



Window Type: Sliding

Window Size

Specify by: Value

Window Size: 10 Minutes (m)

Sliding Interval

Specify by: Value

Sliding Interval: 5 Minutes (m)

Watermark Delay

Specify by: Value

Watermark Delay: 10 Minutes (m)

Use case & Demo

Use Case & Demo

Imagine you started a ride hauling company and need to check if the vehicles are over-speeding. We will create a simple near real-time streaming application to calculate the maximum speed of vehicles every few seconds, while talking about **the concept of window transformation**

Troubleshooting / Self-Service

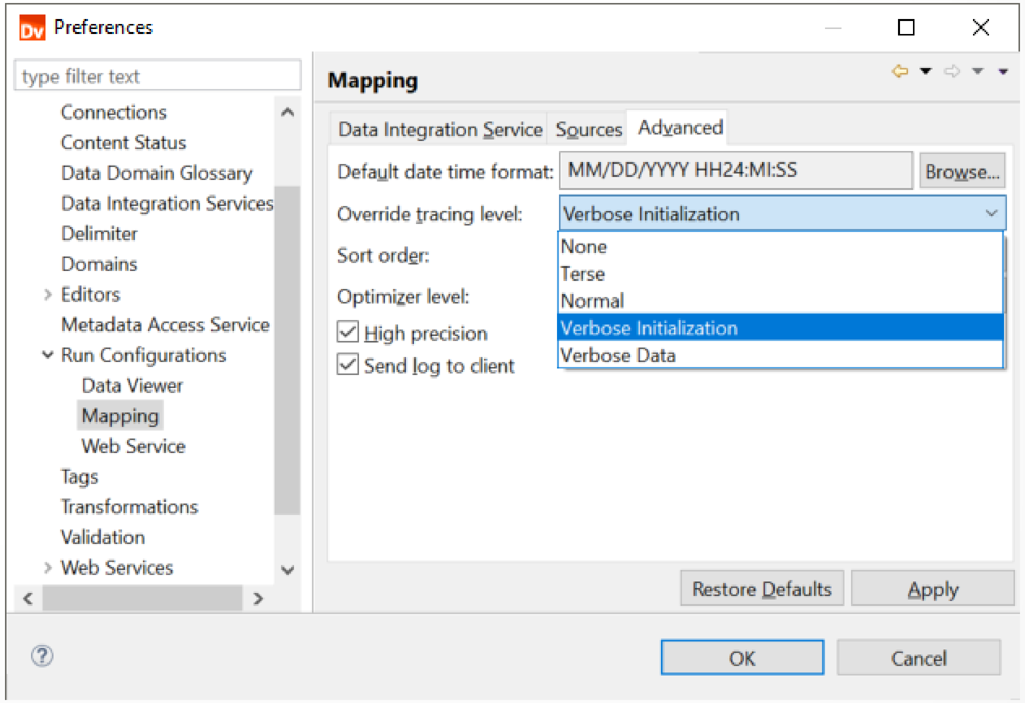
Troubleshooting

Logs

- Mapping log
- Spark application log

Override Tracing – Log level

- Normal - INFO
- Verbose Init - DEBUG [Recommended for debugging]
- Verbose Data - DEBUG



Mapping

Default date time format: MM/DD/YYYY HH24:MI:SS

Override tracing level: Verbose Initialization

Sort order:

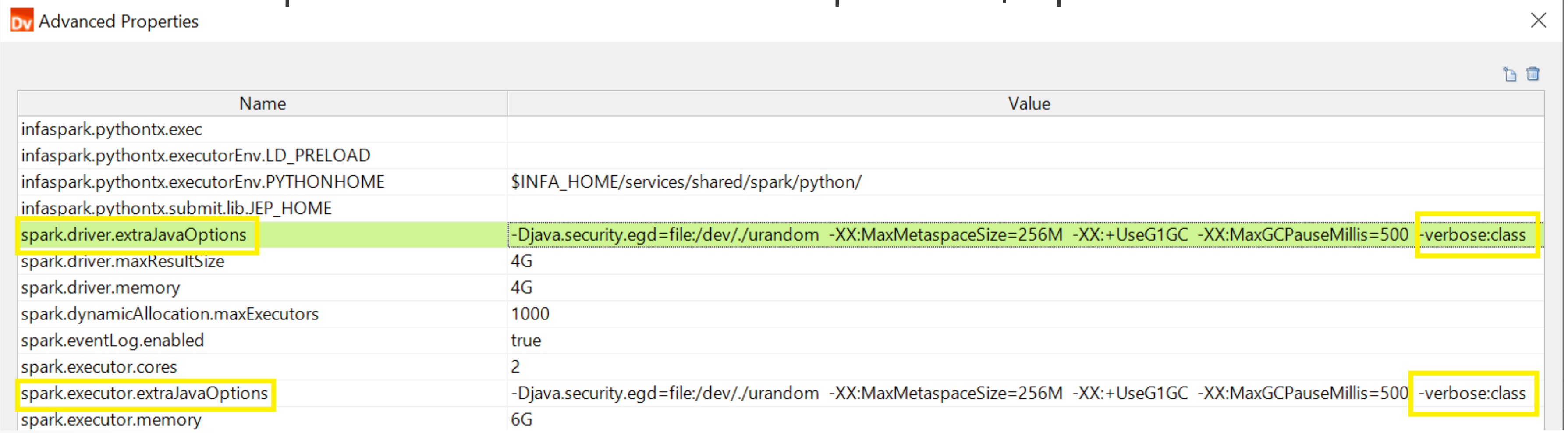
Optimizer level:

☑ High precision

☑ Send log to client

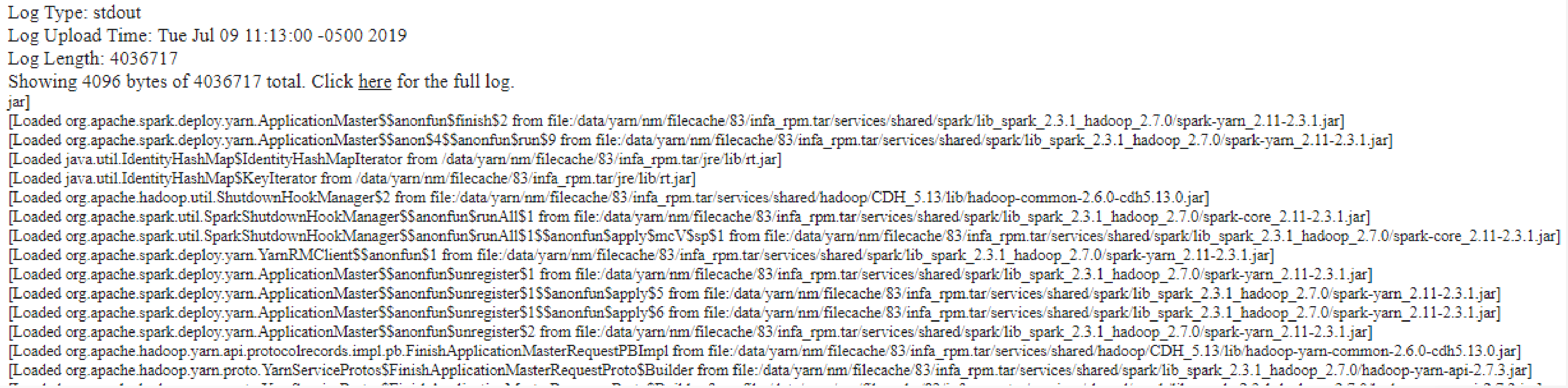
Troubleshooting

spark.driver.extraJavaOptions | spark.executor.extraJavaOptions



Name	Value
infaspark.pythontx.exec	
infaspark.pythontx.executorEnv.LD_PRELOAD	
infaspark.pythontx.executorEnv.PYTHONHOME	$INFA_HOME/services/shared/spark/python/
infaspark.pythontx.submit.lib.JEP_HOME	
spark.driver.extraJavaOptions	-Djava.security.egd=file:/dev/./urandom -XX:MaxMetaspaceSize=256M -XX:+UseG1GC -XX:MaxGCPauseMillis=500 -verbose:class
spark.driver.maxResultSize	4G
spark.driver.memory	4G
spark.dynamicAllocation.maxExecutors	1000
spark.eventLog.enabled	true
spark.executor.cores	2
spark.executor.extraJavaOptions	-Djava.security.egd=file:/dev/./urandom -XX:MaxMetaspaceSize=256M -XX:+UseG1GC -XX:MaxGCPauseMillis=500 -verbose:class
spark.executor.memory	6G

Hadoop connection



```
Log Type: stdout
Log Upload Time: Tue Jul 09 11:13:00 -0500 2019
Log Length: 4036717
Showing 4096 bytes of 4036717 total. Click here for the full log.
jar]
[Loaded org.apache.spark.deploy.yarn.ApplicationMaster$$anonfun$finish$2 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/spark-yarn_2.11-2.3.1.jar]
[Loaded org.apache.spark.deploy.yarn.ApplicationMaster$$anon$4$$anonfun$run$9 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/spark-yarn_2.11-2.3.1.jar]
[Loaded java.util.IdentityHashMap$IdentityHashMapIterator from /data/yarn/nm/filecache/83/infa_rpm.tar/jre/lib/rt.jar]
[Loaded java.util.IdentityHashMap$KeyIterator from /data/yarn/nm/filecache/83/infa_rpm.tar/jre/lib/rt.jar]
[Loaded org.apache.hadoop.util.ShutdownHookManager$2 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/hadoop/CDH_5.13/lib/hadoop-common-2.6.0-cdh5.13.0.jar]
[Loaded org.apache.spark.util.SparkShutdownHookManager$$anonfun$runAll$1 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/spark-core_2.11-2.3.1.jar]
[Loaded org.apache.spark.util.SparkShutdownHookManager$$anonfun$runAll$1$$anonfun$apply$mcV$sp$1 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/spark-core_2.11-2.3.1.jar]
[Loaded org.apache.spark.deploy.yarn.YarnRMClient$$anonfun$1 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/spark-yarn_2.11-2.3.1.jar]
[Loaded org.apache.spark.deploy.yarn.ApplicationMaster$$anonfun$unregister$1 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/spark-yarn_2.11-2.3.1.jar]
[Loaded org.apache.spark.deploy.yarn.ApplicationMaster$$anonfun$unregister$1$$anonfun$apply$5 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/spark-yarn_2.11-2.3.1.jar]
[Loaded org.apache.spark.deploy.yarn.ApplicationMaster$$anonfun$unregister$1$$anonfun$apply$6 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/spark-yarn_2.11-2.3.1.jar]
[Loaded org.apache.spark.deploy.yarn.ApplicationMaster$$anonfun$unregister$2 from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/spark-yarn_2.11-2.3.1.jar]
[Loaded org.apache.hadoop.yarn.api.protocolrecords.impl.pb.FinishApplicationMasterRequestPBImpl from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/hadoop/CDH_5.13/lib/hadoop-yarn-common-2.6.0-cdh5.13.0.jar]
[Loaded org.apache.hadoop.yarn.proto.YarnServiceProtos$FinishApplicationMasterRequestProto$Builder from file:/data/yarn/nm/filecache/83/infa_rpm.tar/services/shared/spark/lib_spark_2.3.1_hadoop_2.7.0/hadoop-yarn-api-2.7.3.jar]
```

Spark application log

Troubleshooting

- If you are upgrading from 10.2.1 -> 10.2.2 & later release, recreate the Data Objects for message header support.
- Common Issues

References

- Data Engineering Streaming User Guide
- Structured Streaming
- Product Availability Matrix
- Data Engineering community forum
- Release notes

Q&A

Thank You

Informatica™