Writing applications at the Edge with OSIsoft Edge Data Store

Frank Gasparro - Engineering Group Lead

Sean Driscoll – Software Developer

Who's Who

- **Sean Driscoll**

sdriscoll@osisoft.com

Software Developer
OSIsoft, LLC

- **Frank Gasparro**

fgasparro@osisoft.com

Engineering Group Lead
OSIsoft, LLC

Agenda and Goals

- Overview of Edge Data Store's Roadmap
- Provide guidance on basic setup and configuration
- Provide a background understanding of how to write data into Edge Data Store
- Overview of our REST API and Edge Data Store's capabilities

Edge Data Store:
Living on the Edge

Edge Data Store

Edge Data Store

Assets

Persistent Storage

Self-Healing

Application Platform

- Cross Platform (Windows, Linux)
- Multiple Data Ingress Options
- Developer API for custom apps and analytics
- Egress to PI Server and OSIsoft Cloud Services
- Developed, Sold, and Supported by OSIsoft
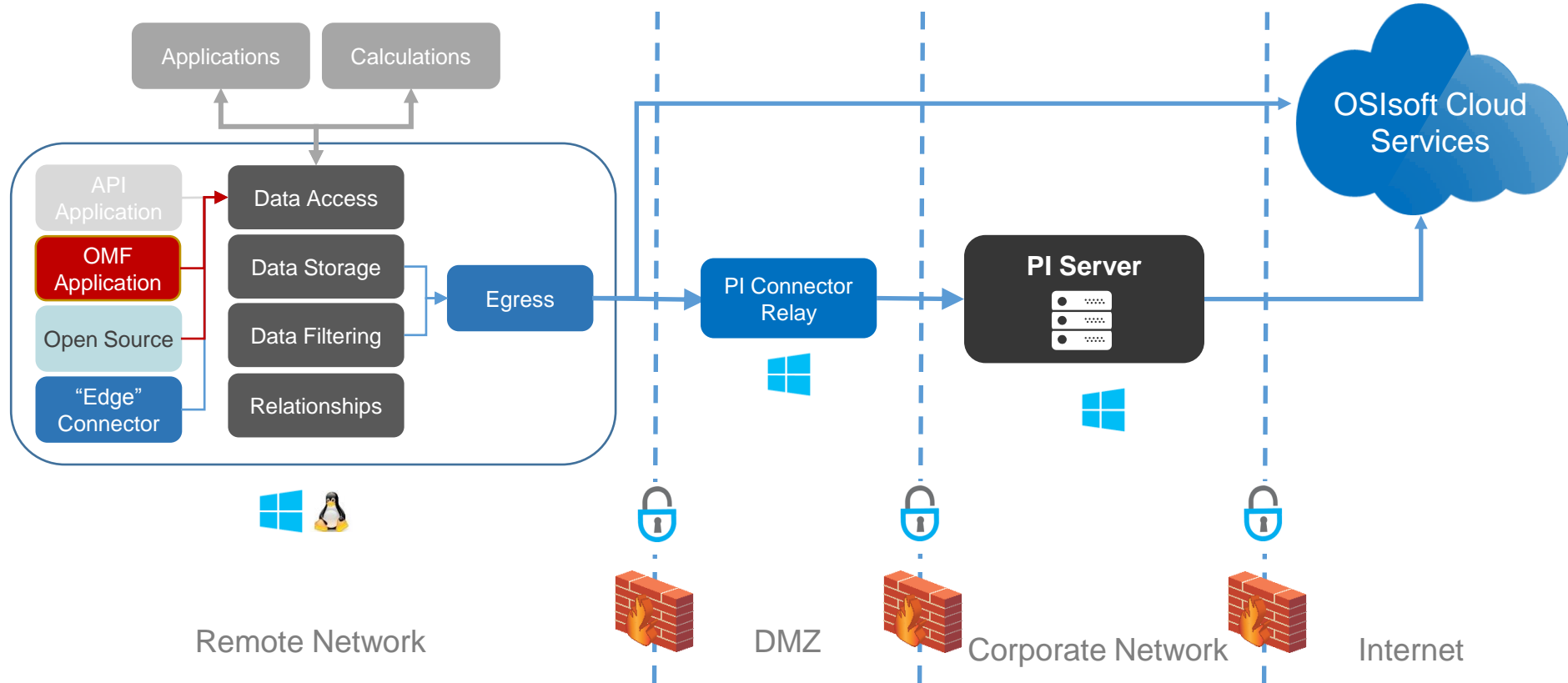
Edge Data Store Overview

- A new OSIsoft product built for remote, unmanned deployments in harsh environments, on lightweight, Windows and Linux devices
- A "sequential" database
  - Anything that can be ordered can be stored
- Shares the same storage technology and API as OSIsoft Cloud Services
- Product features
  - Simple and Complex data types
  - Multiple indexes
    - Non-time series
    - Compound
    - Secondary
  - No distinction between future and past data
  - Unbounded Event Size
  - Ingress via OMF, API, and selected PI Connectors
  - Egress to PI Servers and/or OSIsoft Cloud Services
  - High precision timestamps (100ns)
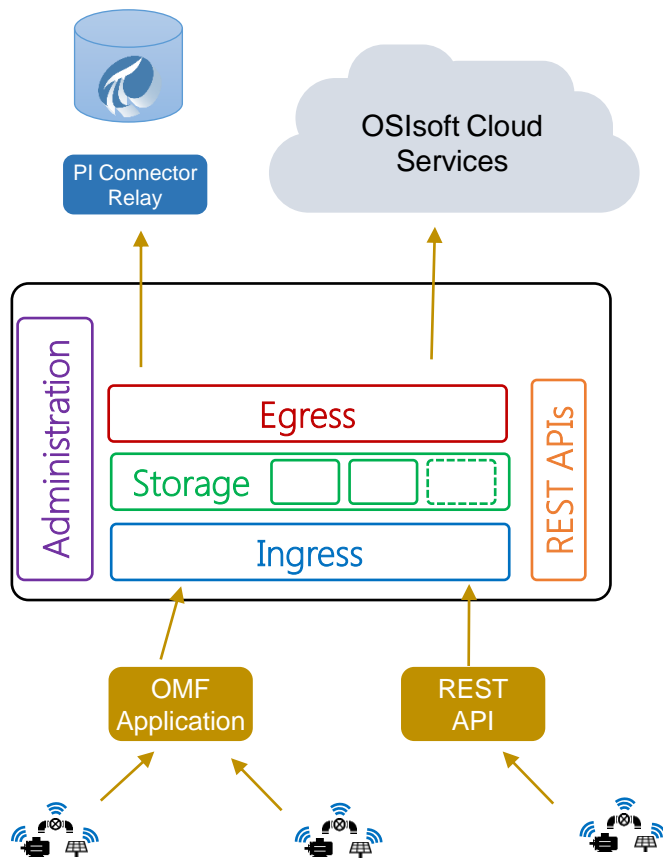
Edge Data Store Hardware Requirements

- Targeting gateway devices
  - 1 GB RAM
  - ARM or x64 CPU, 1.2 GHz quad core
  - Internal Storage
  - Linux (Ubuntu 16.04)
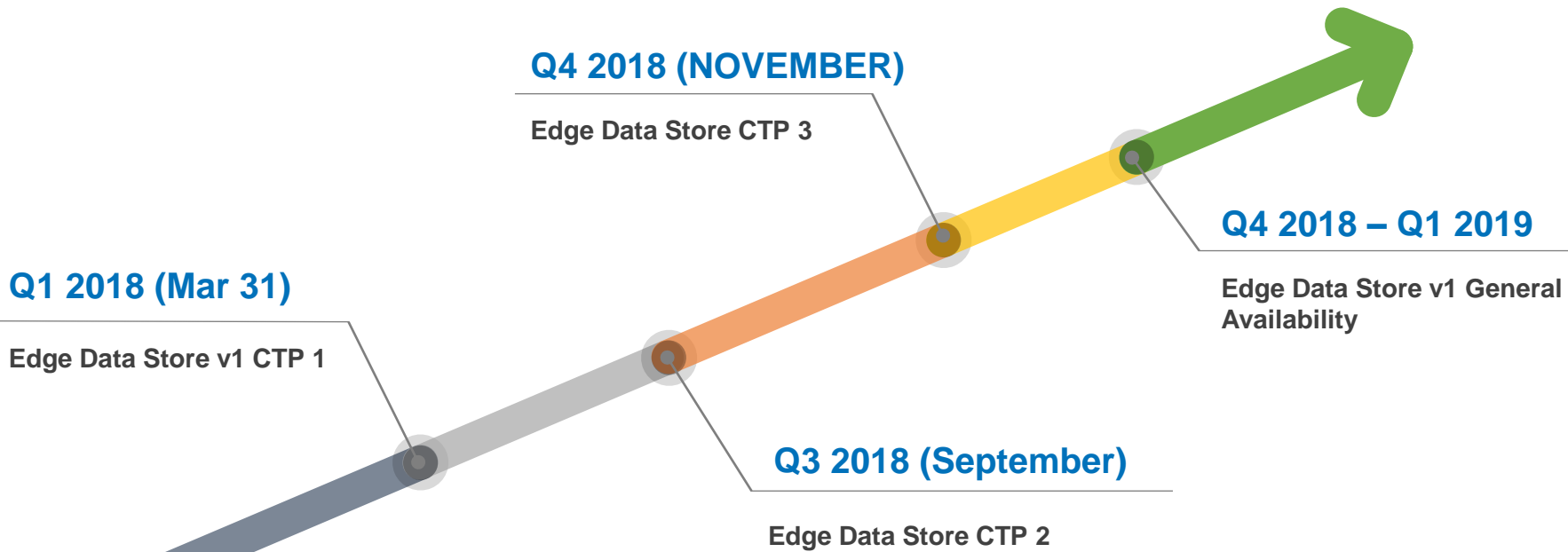 - Windows - Windows 10 or higher

Edge Data Store Version 1 Architecture

Applications   Calculations

API Application
OMF Application
Open Source
"Edge" Connector

Data Access
Data Storage
Data Filtering
Relationships

Egress

PI Connector Relay

PI Server

OSIsoft Cloud Services

Remote Network

DMZ

Corporate Network

Internet

Edge DS – CTP 2 Capabilities

Edge Data Store Development Timeline

Q4 2018 (NOVEMBER)

Edge Data Store CTP 3

Q1 2018 (Mar 31)

Edge Data Store v1 CTP 1

Q4 2018 – Q1 2019

Edge Data Store v1 General Availability

Q3 2018 (September)

Edge Data Store CTP 2

GA = General Availability (released to market)
CTP = Customer Technology Preview (not ready for release)

How do I get started?

- Where do we get the Edge DS Bits?
  - CTP 2 is by invitation only
  - https://techsupport.osisoft.com/Downloads/All-Downloads/All-Groups/All-Products/All-Categories/Current-Version/edge/
- Where do I find code samples and libraries
  - https://github.com/osisoft/
- Documents
  - http://osisoft-edge.readthedocs.io/en/latest/EdgeDS_Admin.html

Edge Data Store Setup and Configuration 101

Deployment Options
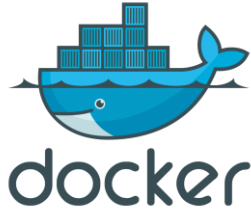
- Private Registry
  - Edgedev.azurecr.io
  - Get password and ID from: EdgeDSSupport@osisoft.com
  - Linux based images only

- BITS Install
- Download
  - Technicalsupport.osisoft.com
- Windows
- Linux x64
- Linux ARM (32)

DEMO

- Docker image
  - Get it running
  - Install bits
  - Show bits running

- Linux Bits – What do I need to do to get them running?

Edge Data Store Admin API

Frank Gasparro

Edge DS Admin API – What will we cover?

- Swagger Documented API
- Scripting Example

Edge Data Store
Data Ingress with OMF

Frank Gasparro

OSIsoft Message Format

OMF Application

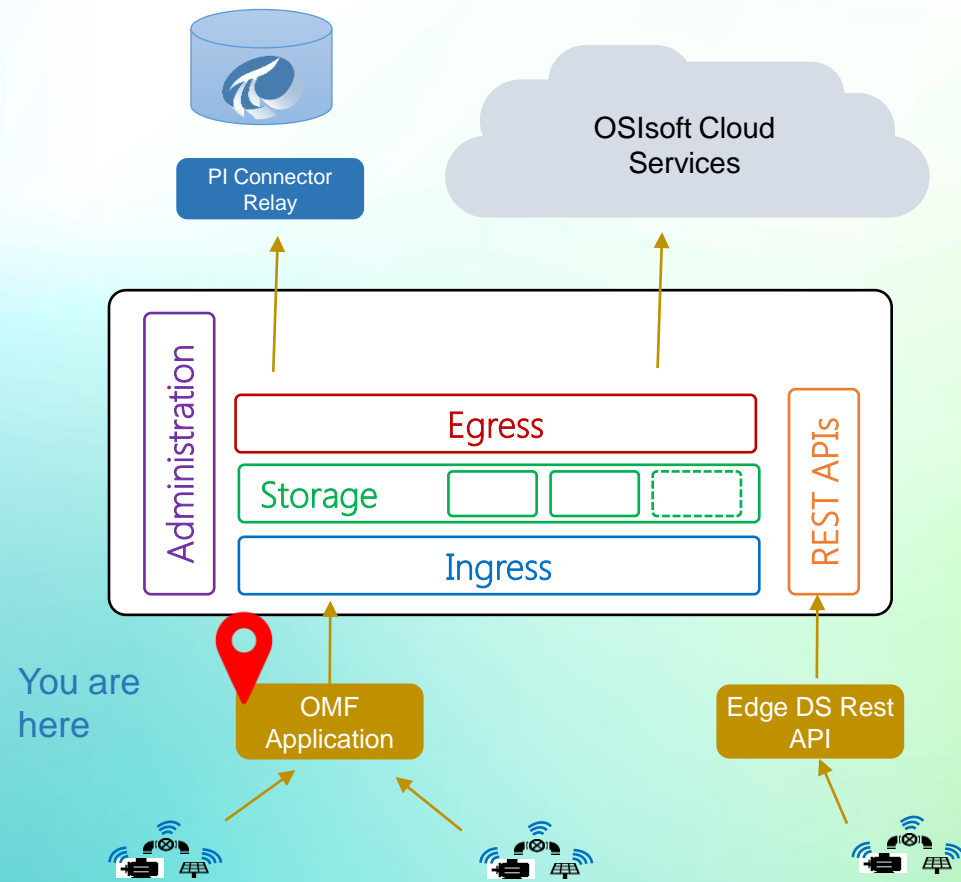Sensors

Maximum Developer Flexibility

Lightweight Footprint

Agnostic to Environment

- Any Hardware
- Any Operating System
- Any Development Tools
- Data Ingress to Edge Data Store, PI Server and OSIsoft Cloud Services
- Finished Product Developed, Sold and Supported by 3rd Parties

EDS and OMF

PI Connector Relay

OSIsoft Cloud Services

Administration

Egress

Storage

REST APIs

Ingress

You are here

OMF Application

Edge DS Rest API

Getting data into Edge Data Store

- Provide the developer with good options for data ingress:

  - OMF Ingress – common standard across OSIsoft
  - REST Ingress – similar surface area as OCS
  - REST API still used for Queries, Management

- These options protect your investment in app development

- Both Ingress options are straightforward, approachable with different languages

- The programming model is similar in either case…

Ingress - Code Pattern

1. Create Tenant, Namespace (if desired)
    - Use REST API

2. Create Type

3. Create Streams (Containers)

4. Send data associated with stream

Ingress – Code Pattern

- Objects are passed as JSON Strings via HTTPS requests

- Helpful to use library for JSON generation – e.g. in .Net
  - Newtonsoft Json.NET
  - OMF Ingress – OSIsoft.Parse.OMFGeneration
  - REST API – OSIsoft Qi Client Library

- App must provide required HTTP header values
  - Ingress always expects "producertoken"
  - OMF – message type, action, etc. expected

Complex and Non-Time Index Data Type Example

Simple Data Type (individual sensors) Indexed by Time

{"Latitude": "29.7817",
"Timestamp": "2017-12-01"}
{"Longitude": "-95.6112",
"Timestamp": "2017-12-01"}
{"Heading"; "42" ,
"Timestamp": "2017-12-01"}
{"Heart-rate": "143",
"timestamp": "2017-12-01"}
{"Temperature": "98.6",
"Timestamp": "2017-12-01"}
{"Count": "10000",
"Timestamp": "2017-12-01"}

Complex Data Type ("fitness band") Indexed by Count

{"Latitude": "29.7817",
"Longitude": "-95.6112",
"Heading"; "42" ,
"Heart-rate": "143",
"Temperature": "98.6",
"Timestamp": "2017-12-01",
🔑 "StepCount": "10,000"}

Edge DS Overview

PI Connector Relay

OSIsoft Cloud Services

Administration

Egress

Storage

Ingress

REST APIs

You are here

OMF Application

Edge DS Rest API
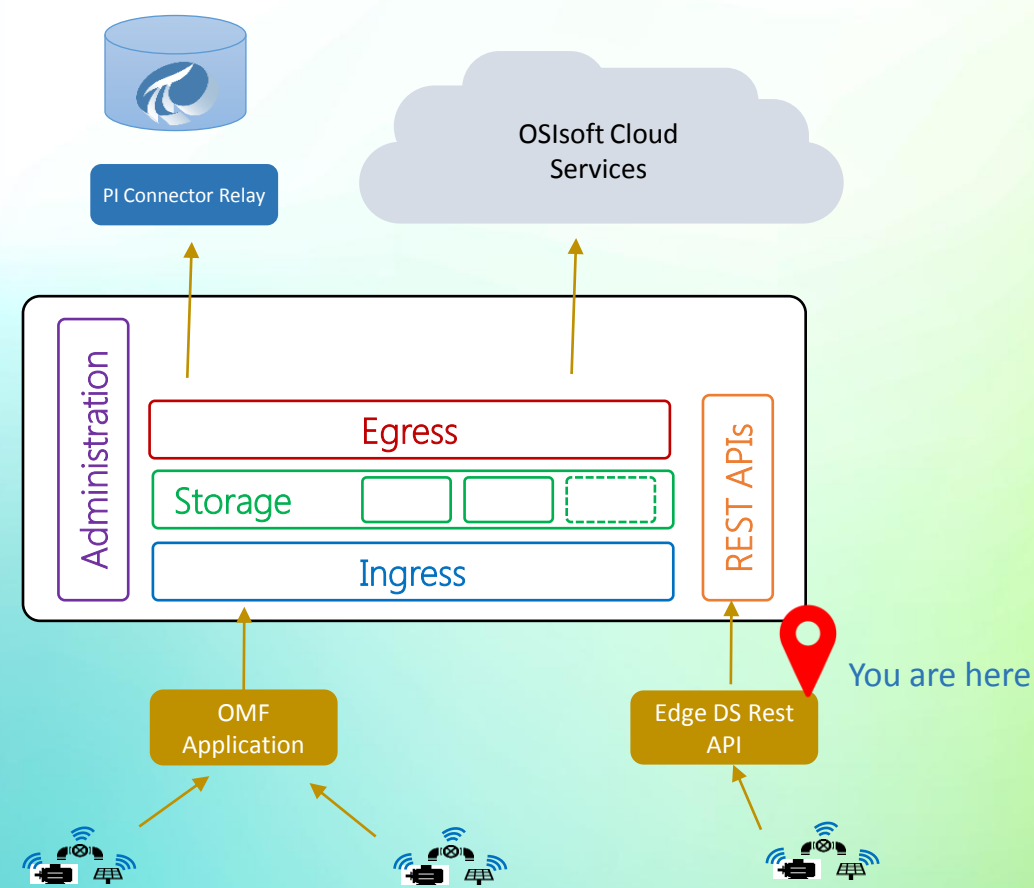
What are we going to cover?

- Performing Queries against Streams
- Edge Data Store Specific Methods
- Resiliency
- Sending data to other OSIsoft systems
  - What exactly is Egressed?
  - Configuring EgressTarget
  - Configuring EgressRules

But 1st a little about Edge and OCS Types and Streams

Data Structure: Types – Simple Cases

Type: Simple Double

Timestamp: DateTime

Value: Double

Quality info: Boolean

Type: Simple Integer

Timestamp: DateTime

Value: Integer

Quality info: Boolean

Data Structure: Types – Complex Cases

Type: **Platform pressure**

- **Timestamp**: DateTime 🔑
- **Pressure**: Double
- **Depth**: Double
- **Area code**: Integer
- **Quality info**: String
- **Edited**: DateTime?

Type: **Batch measurement**

- **Lot number**: Integer 🔑
- **pH**: Single
- **Color**: String
- **Weight**: Double
- **Quality info**: String
- **Tested**: DateTime?

Data Storage: Streams (instances of a Type)

Type: Platform pressure

Timestamp: DateTime

Pressure: Double

Depth: Double

Area code: Integer

Quality info: String

Edited: DateTime?

Stream: **Platform 434-E**

Stream: **Platform 709-G**

Stream: **Platform 432-N**

Type: **Platform pressure**

Name: Platform 432-N

Description: **Fronts…**

Tags: **Drilling, EU,…**
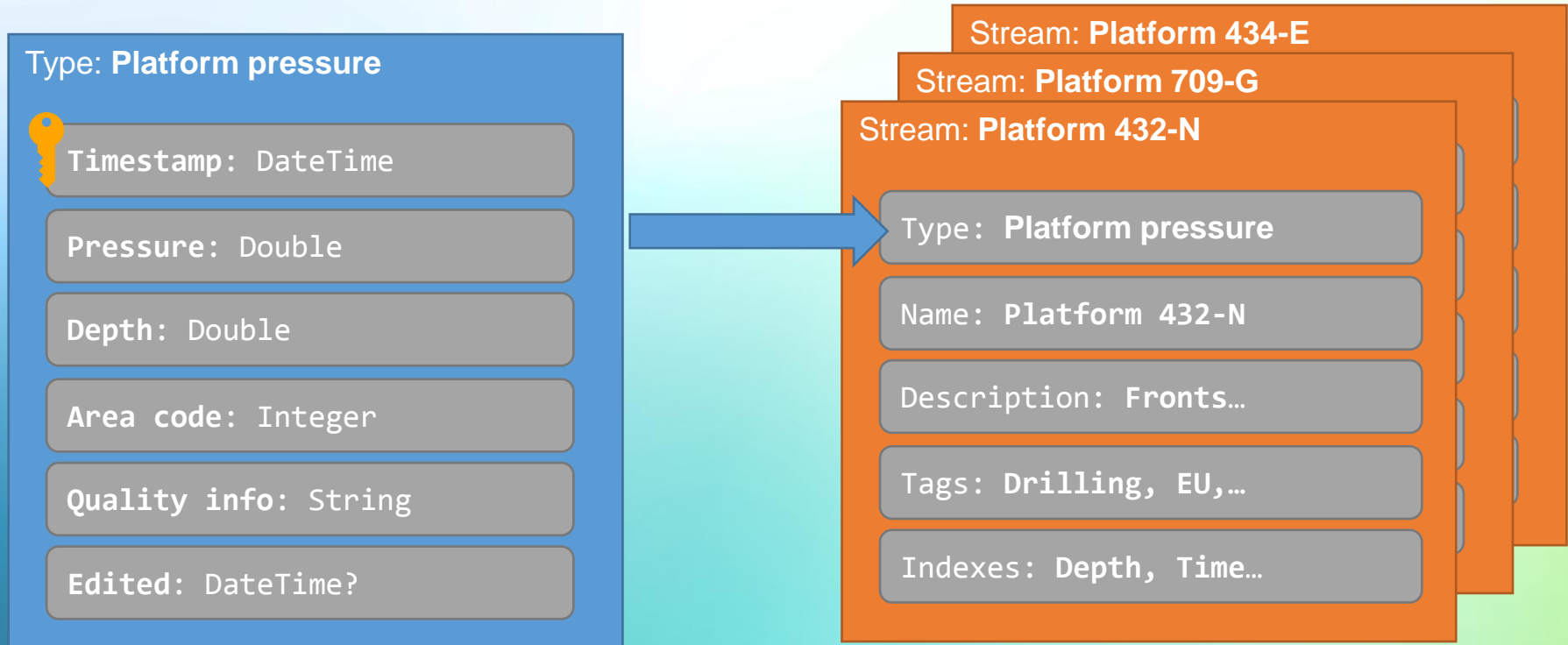
Indexes: **Depth, Time…**

Data Ingress: Streaming data

Stream: Platform 432-N

Type: **Platform pressure**

Name: **Platform 432-N**

Description: **Fronts…**

Tags: **Drilling, EU,…**

Indexes: **Depth, Time…**

New event

New event -
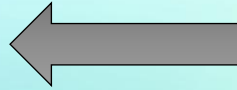
New event -

New event -

New event

Stream: **Platform 432-N**

Timestamp: 24-Oct-2017…

Pressure: **134.0254**

Depth: **-256.12**

Area code: **254**

Quality info: **Quest…**

Edited: **null**