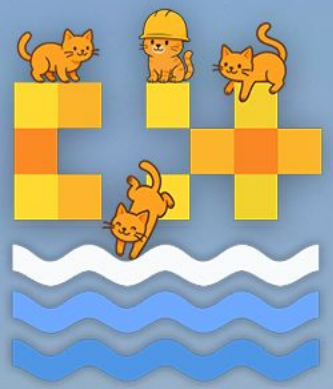


Building the Backbone for Market Data in Modern C++

Jody Hagins,
Robert Leahy
& Gianluca Delfino

2025

Building the Backbone for Market Data in Modern C++

Jody Hagins – Director, Software Engineering
jody.hagins@lseg.com
coachhagins@gmail.com

Gianluca Delfino – Director, Software Engineering
gianluca.delfino@lseg.com
gianluca.delfino@gmail.com

Robert Leahy – Lead Software Engineer
robert.leahy@lseg.com
rleahy@rleahy.ca

LSEG

What we do in the “Real Time **Ultra** Direct” Team

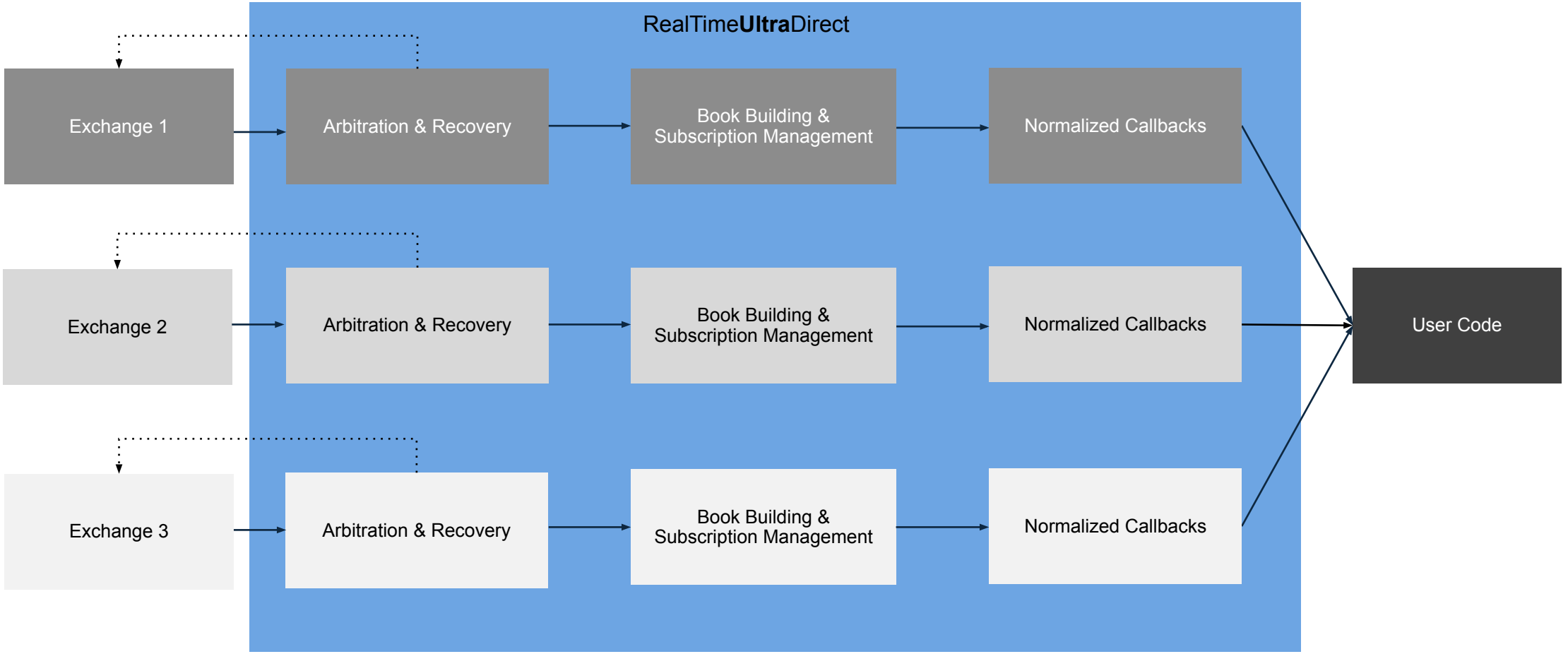
...and how we do it!

LSEG

What is RealTime**Ultra**Direct

- C++ library that can handle millions of messages per second
- Mission: democratize HFT
- We provide *Normalization* for hundreds of feeds over many exchanges
- Low barrier of entry, easy to set up and start using.
- Comes with Batteries included:
  - built for every Linux platform/Compiler combination
  - C++/Java bindings
  - Performance analysis tools
- Extensible:
  - Buy single feed-handlers / pay for what you use
- “Extra mile” Support:
  - Direct access to developers for technical issues

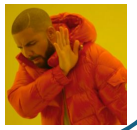
One “Normalized” Interface to rule them ALL



One “Normalized” Interface to rule them ALL

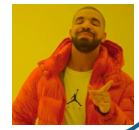
example “RAW”

ComplexInstrumentDefinition
SingleSideUpdate
TwoSideUpdate
TopTrade
AddOrderLong
AddOrderShort
AddOrderExpanded
OrderExecuted
ModifyOrderLong
ModifyOrderShort
DeleteOrder
TradeLong
TradingStatus
AuctionUpdate
AuctionCancel
AuctionTrade
…



Normalized

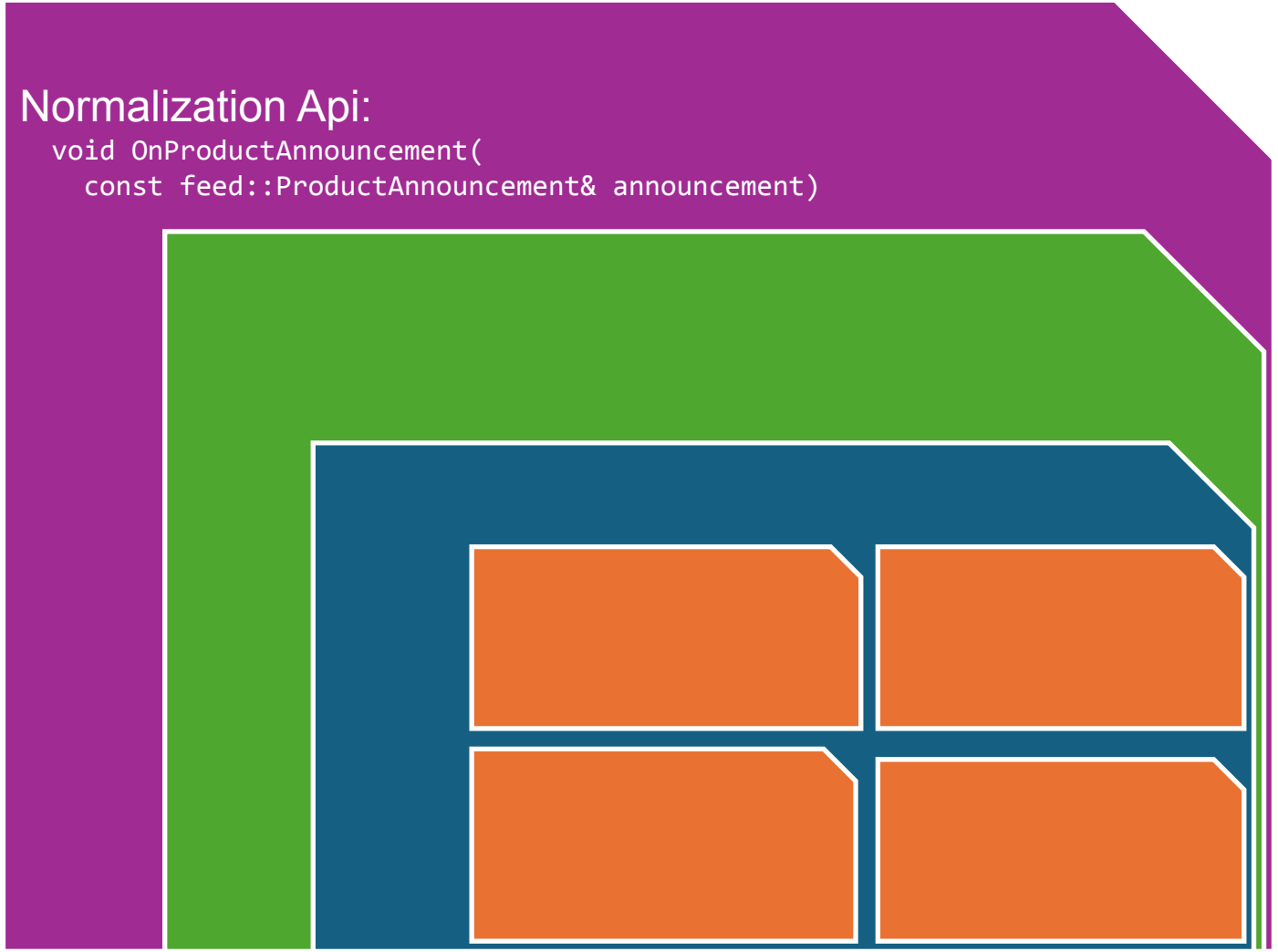
ProductAnnouncement
ProductStatistics
AddOrder
ModifyOrder
CancelOrder
PriceLevelUpdate
OrderImbalance
AuctionSummary
ProductStatus
FeedStatus
MarketStatus
BBOQuote
NBBOQuote
Trade
TradeCorrection
IndexUpdate
…



How do we do it? (API)

A good API is shaped like an Onion

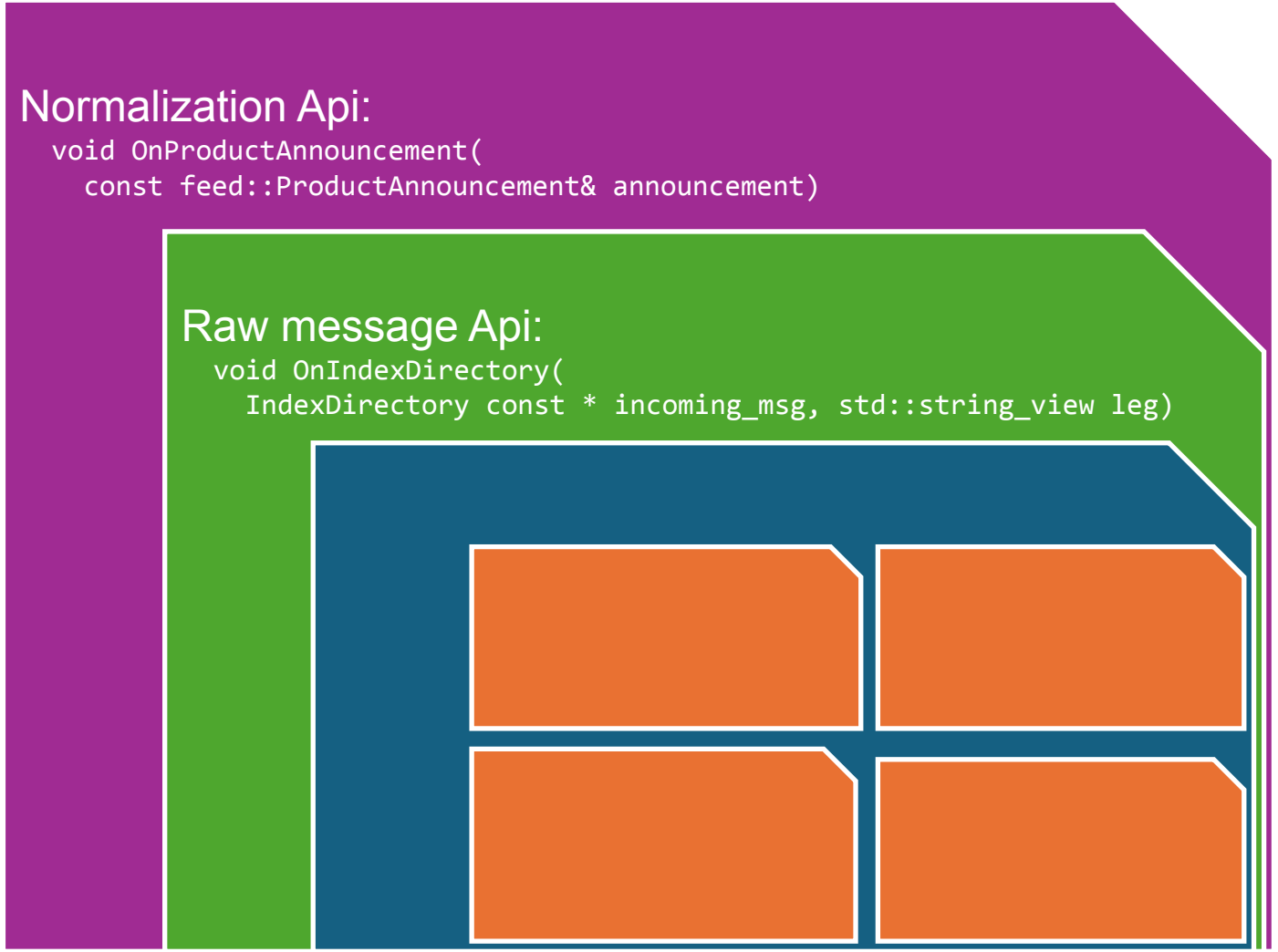
- The external layer is *Normalized* Messages Callbacks



How do we do it? (API)

A good API is shaped like an Onion

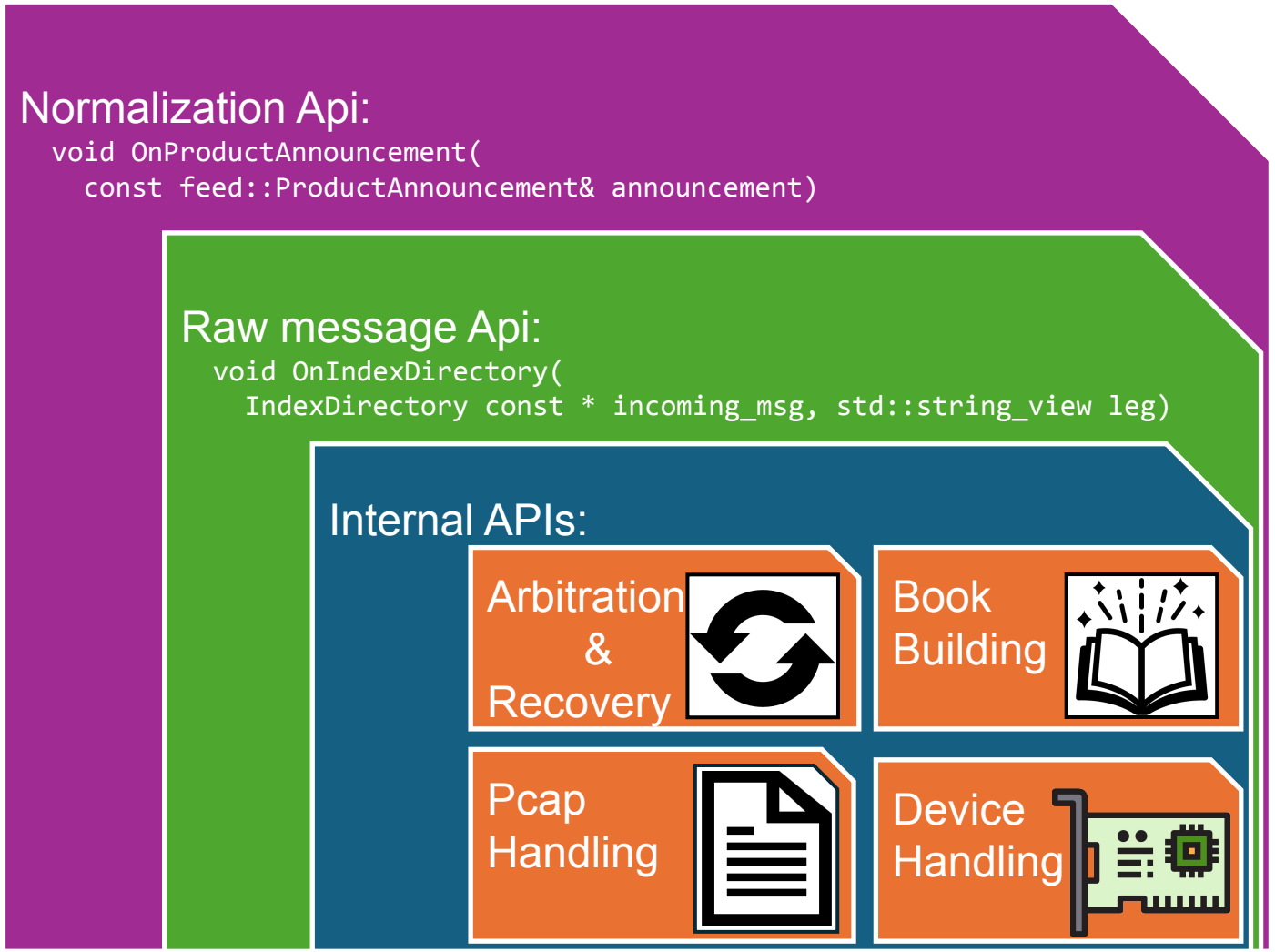
- The external layer is *Normalized* Messages Callbacks
- Optionally users can access “Raw” Messages



How do we do it? (API)

A good API is shaped like an Onion

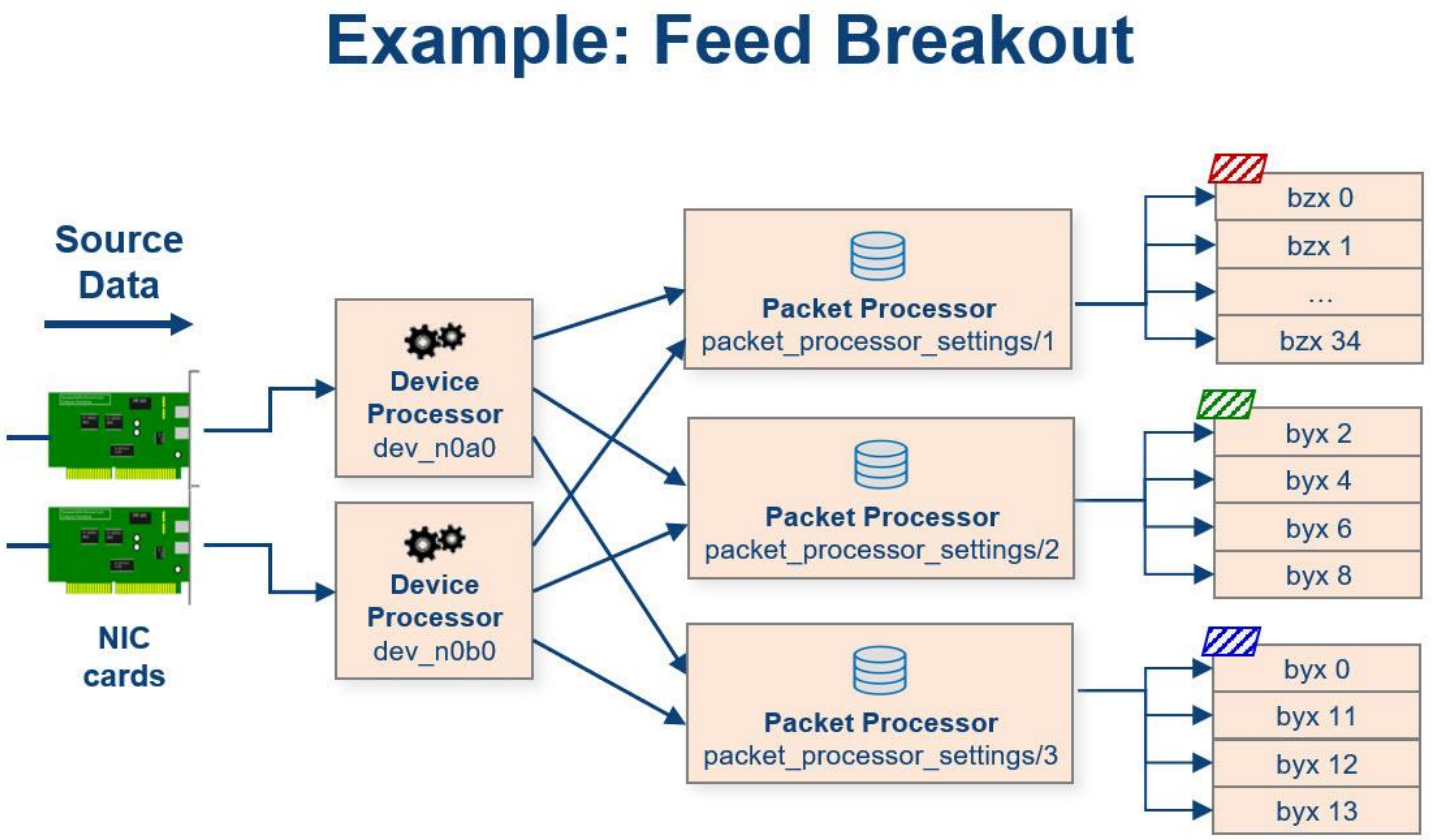
- The external layer is *Normalized* Messages Callbacks
- Optionally users can access “Raw” Messages



How do we do it? (API)

Fully Configurable:

- Devices support *ef_vi* kernel bypass
- Packet processed on specified *threads*



How do we do it? (API)

Our Tech Stack

- More than 2.5 Million lines of C++
- Java bindings are not included in this breakdown
- C++17

```
$ cloc .
  15270 text files.
  14398 unique files.
    928 files ignored.


github.com/AlDanial/cloc v 2.04  T=51.72 s (278.4 files/s, 85414.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                           6313         192258         113575        1654211
C/C++ Header                  6296         158230         320269         962718
XML                            379           9548           7466         453605
JSON                            80              0              0         303255
Text                            63          13468              0         103375
CSV                             15           1021              0          44702
CMake                          937           3946           6215          22415
Python                          81           2315            610          10601
XSD                             17            151             55           6213
JavaScript                      18            318            768           6122
TypeScript                      40            768           1329           5490
Markdown                        48            852              1           2663
Bourne Shell                    28            281            113           1384
PHP                             30            193            398           1005
SCSS                             7             49             39            996
Pascal                          14            191            999            840
CSS                              6             78             33            717
SVG                              1              0              0            685
YAML                             3              7             14            634
HTML                            11             16             29            310
Dockerfile                       5             53             30            180
vim script                       1             14              8             45
diff                             2              8             25             37
Visual Studio Solution           1              1              1             18
C                                1              1              0              3
INI                              1              0              0              2
-------------------------------------------------------------------------------
SUM:                         14398         383767         451977        3582226
-------------------------------------------------------------------------------
```

How do we do it? (CI)

Our CI for every commit:

- clang18-ccache
- gcc-14-ccache
- devtoolset-7-ccache

Overnight:

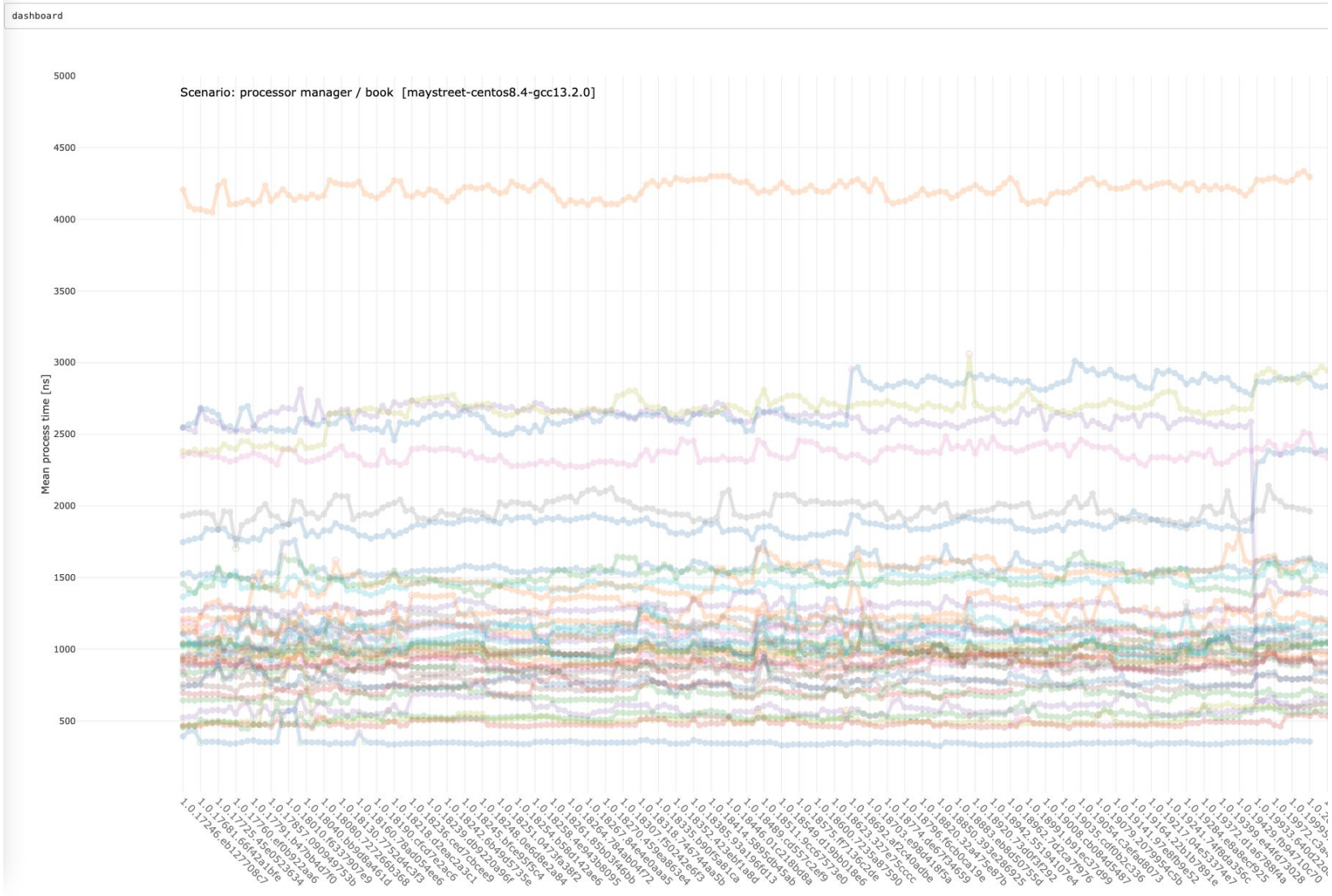
- msan
- tsan
- ubsan
- asan-lsan
- clang-tidy
- memcheck

tipi.build by EngFlow

Performance Monitoring

Continuous monitoring

- Every build is tested with a set of thorough integration tests on production hardware.

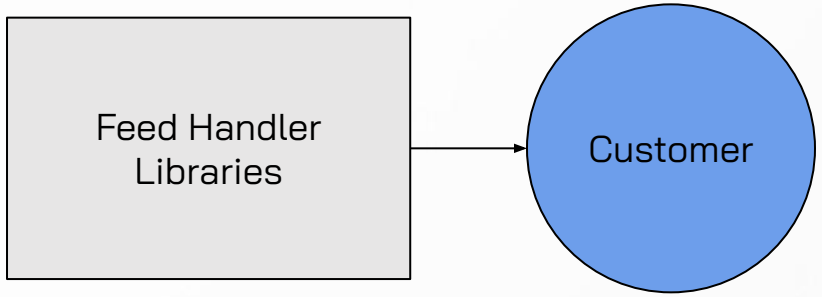


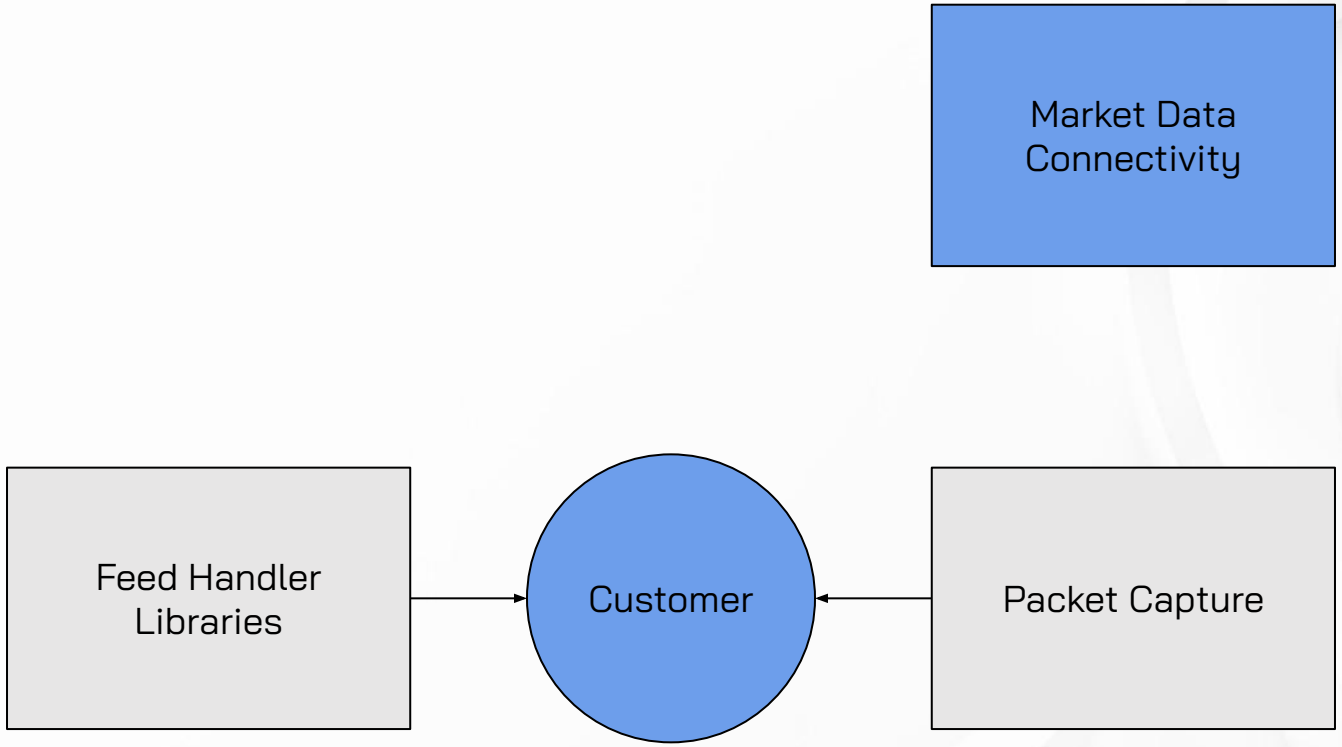
What is next

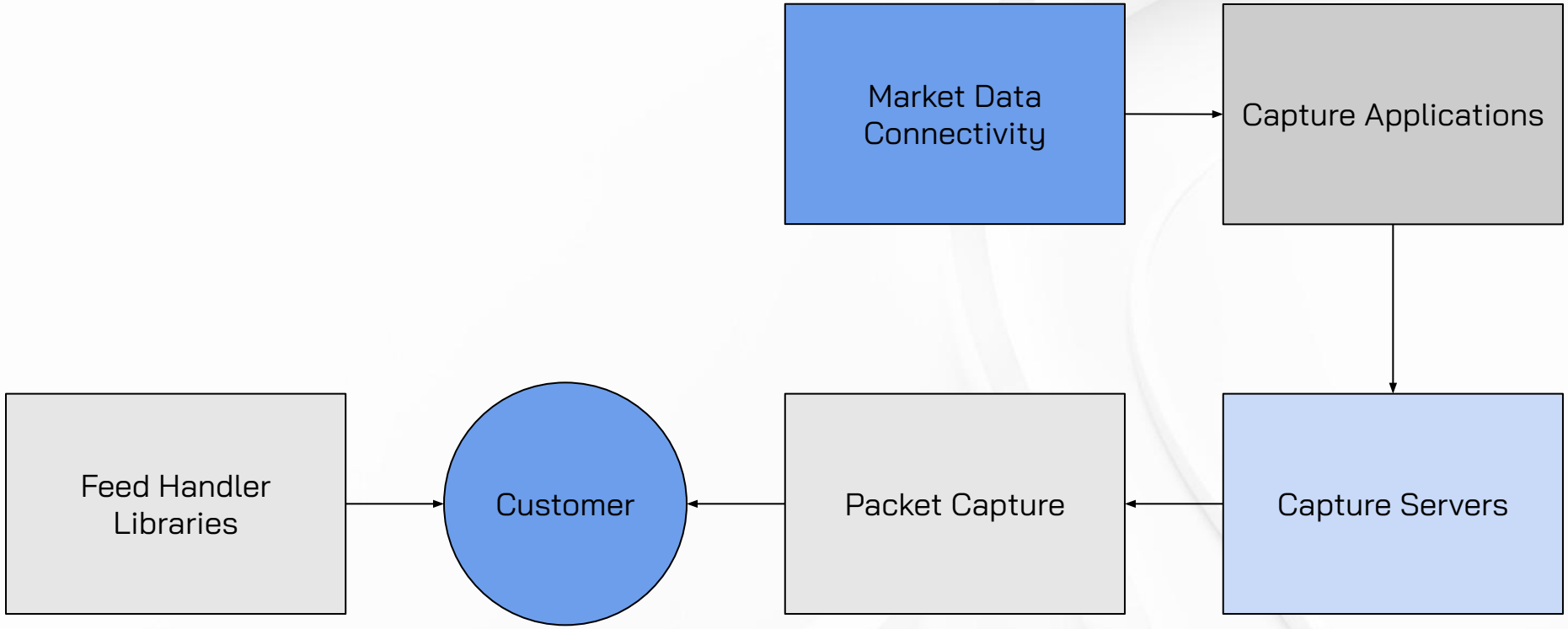
- Better automatic Normalization Code generation
- Customization Options for Normalization (different “flavours”)
- C++20/23 (**Concepts**!)
- Moar faster

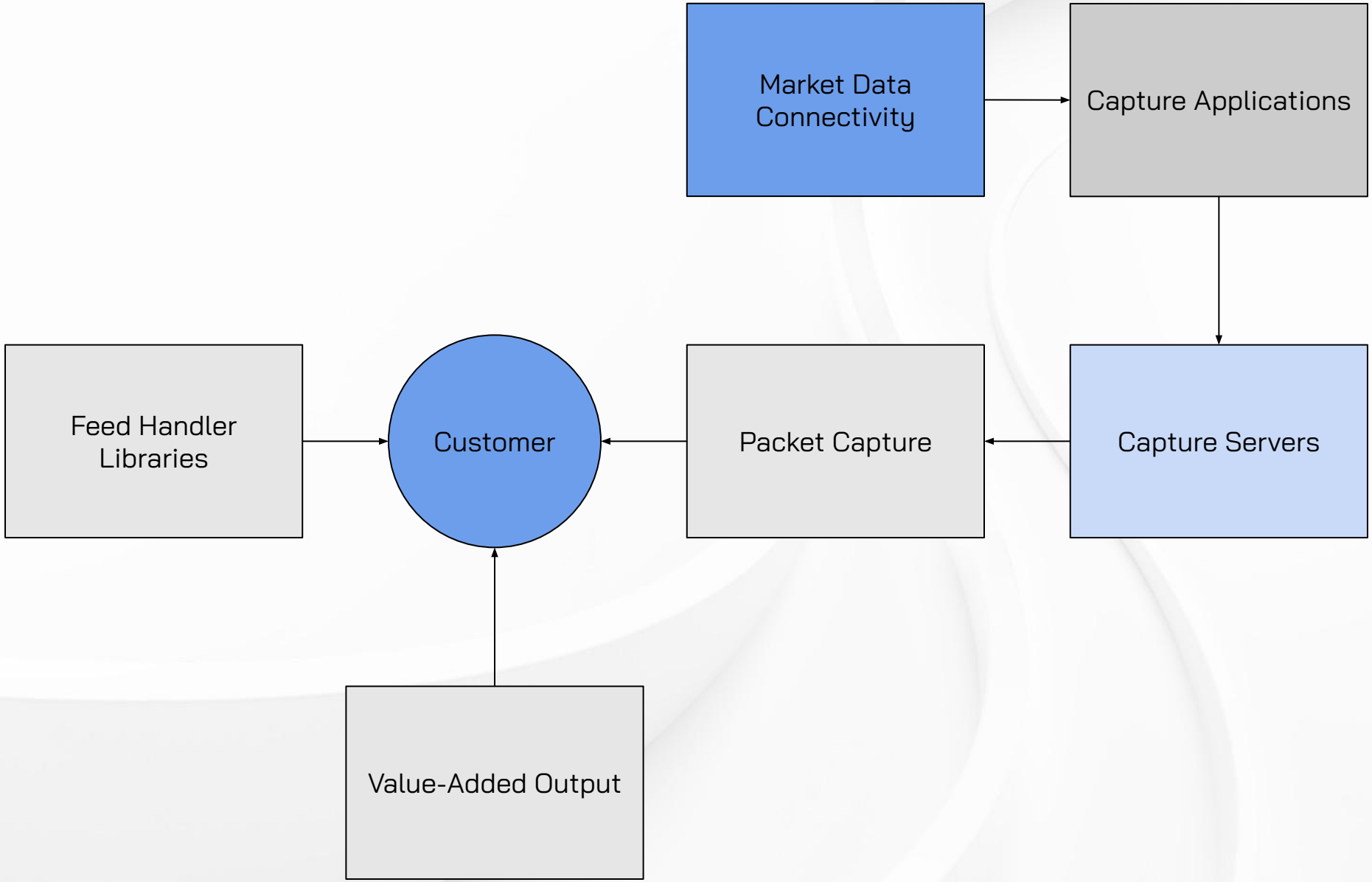
Tools Team

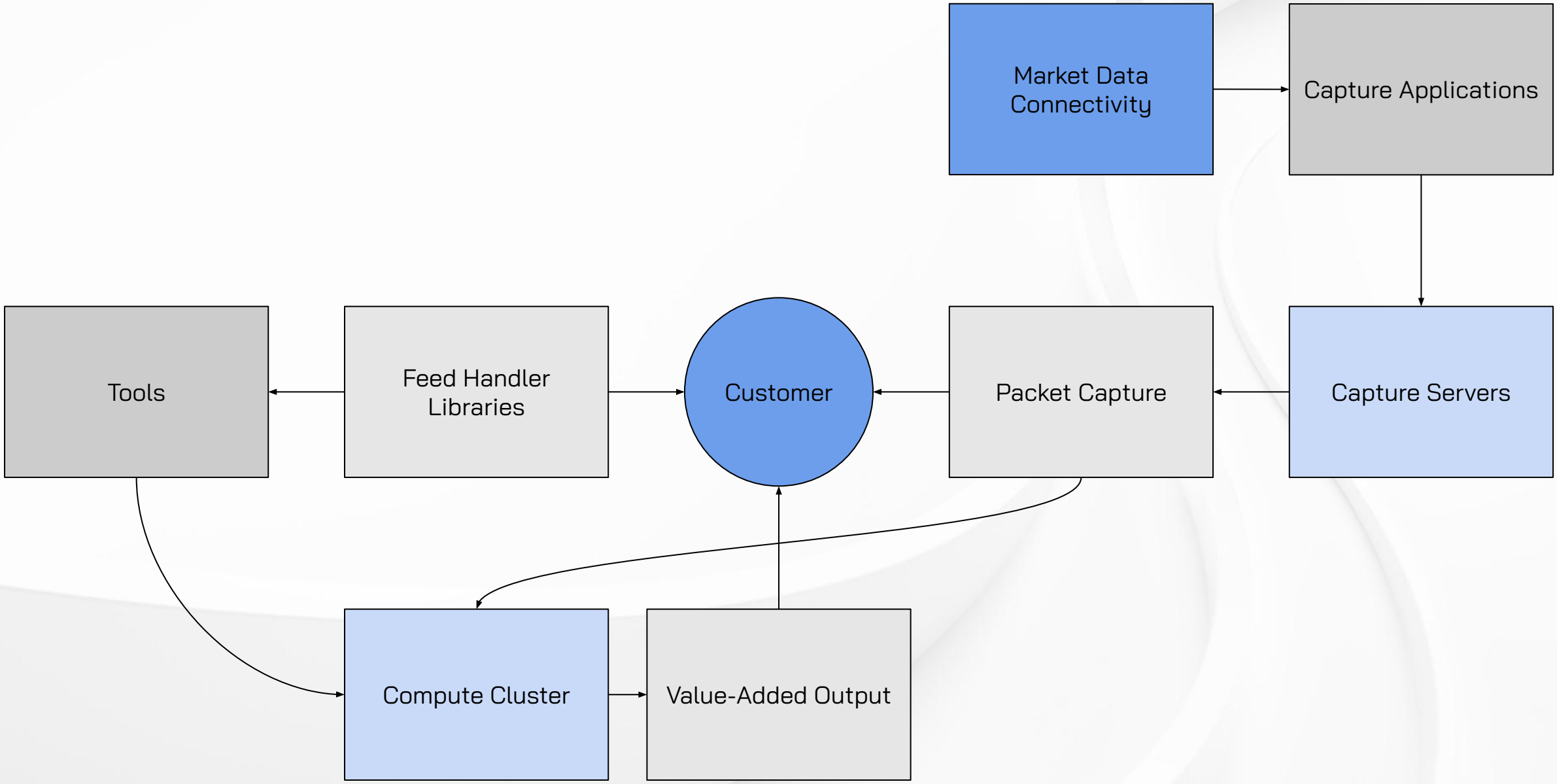
LSEG

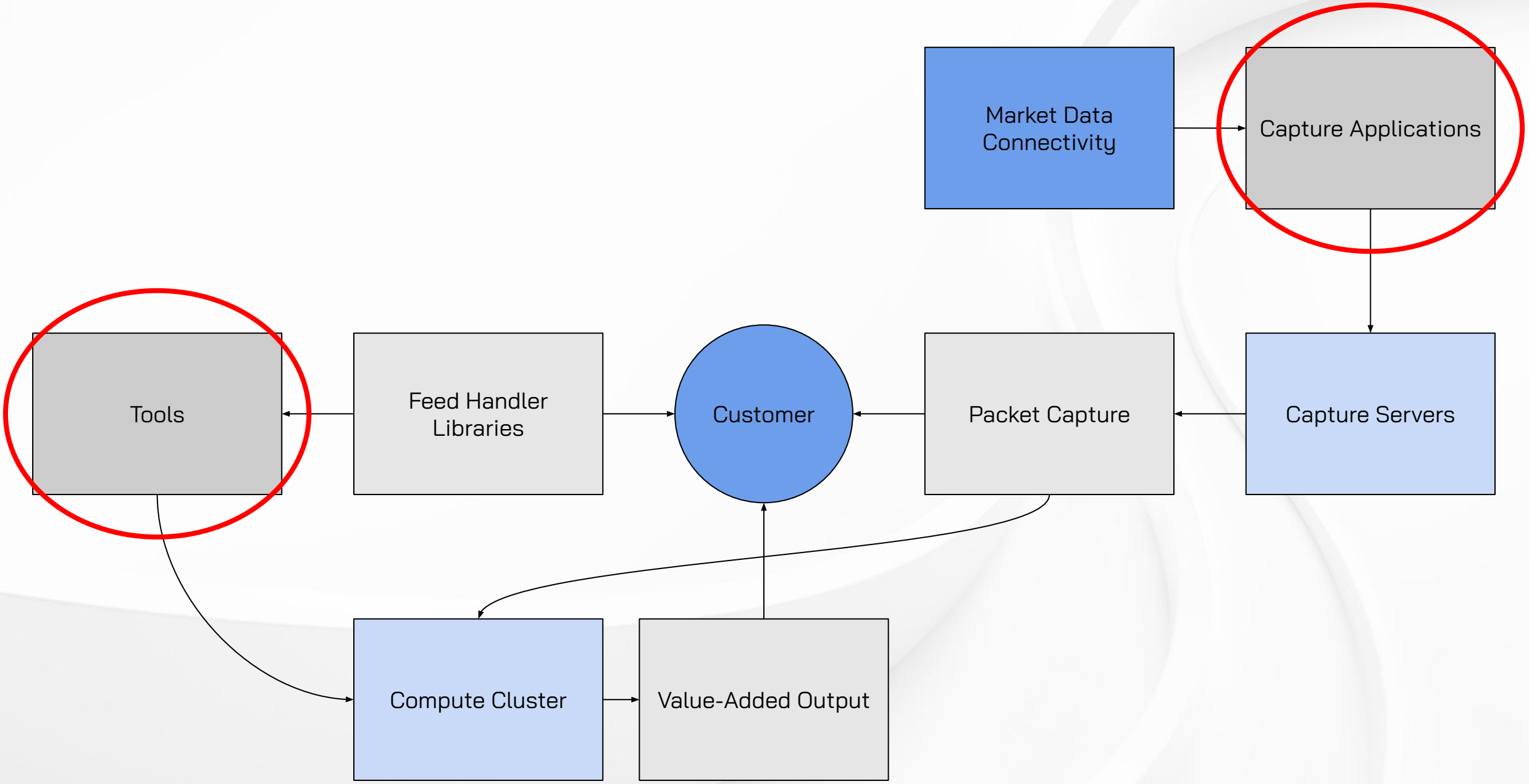

Feed Handler
Libraries
Customer


Market Data Connectivity
Feed Handler Libraries
Customer
Packet Capture


Market Data Connectivity
Capture Applications
Capture Servers
Packet Capture
Customer
Feed Handler Libraries


Market Data Connectivity
Capture Applications
Capture Servers
Packet Capture
Customer
Feed Handler Libraries
Value-Added Output


Market Data Connectivity
Capture Applications
Tools
Feed Handler Libraries
Customer
Packet Capture
Capture Servers
Compute Cluster
Value-Added Output


Market Data Connectivity
Capture Applications
Capture Servers
Packet Capture
Customer
Feed Handler Libraries
Tools
Compute Cluster
Value-Added Output

Breaking it Down

What’re the projects behind the offerings?

Capture

`mstcaprecv`

Captures raw data from network using kernel bypass.

`mstcapwrite`

Writes raw data to raw capture files.

`mstcapmon`

Generates quality metrics about captured data.

`mstsplit`

Post-processes raw capture into client-facing capture offerings.

`replay`

Replays pcaps onto the network for functional testing and simulation.

Breaking it Down

What're the projects behind the offerings?

Capture

`mstcaprecv`

Captures raw data from network using kernel bypass.

`mstcapwrite`

Writes raw data to raw capture files.

`mstcapmon`

Generates quality metrics about captured data.

`mstsplit`

Post-processes raw capture into client-facing capture offerings.

`replay`

Replays pcaps onto the network for functional testing and simulation.

Normalization

`mstnorm`

Transforms market data to various formats (CSV, JSON, Parquet, et cetera) with or without additional aggregate measures.

`mstanalyze`

Pluggable, general purpose processing pipeline.

Breaking it Down

What're the projects behind the offerings?

Capture

`mstcaprecv`

Captures raw data from network using kernel bypass.

`mstcapwrite`

Writes raw data to raw capture files.

`mstcapmon`

Generates quality metrics about captured data.

`mstsplit`

Post-processes raw capture into client-facing capture offerings.

`replay`

Replays pcaps onto the network for functional testing and simulation.

Normalization

`mstnorm`

Transforms market data to various formats (CSV, JSON, Parquet, et cetera) with or without additional aggregate measures.

`mstanalyze`

Pluggable, general purpose processing pipeline.

MDX

Proprietary, distributed, time series database engine.

High Performance Query

REST- and WebSocket-accessible interface for querying market data built on top of MDX.

US 20220044259A1

(19) **United States**

(12) **Patent Application Publication**

Douglas et al.

(10) **Pub. No.: US 2022/0044259 A1**

(43) **Pub. Date: Feb. 10, 2022**

(54) **DATABASE MANAGEMENT SYSTEMS AND METHODS USING DATA NORMALIZATION AND DEFRAGMENTATION TECHNIQUES**

(71) Applicant: **Maystreet, Inc.**, New York, NY (US)

(72) Inventors: **Niall Douglas**, Kerry Pike (IE); **Robert Leahy**, New York, NY (US); **Michael Lehr**, New York, NY (US)

(73) Assignee: **Maystreet, Inc.**, New York, NY (US)

(21) Appl. No.: **16/986,646**

(22) Filed: **Aug. 6, 2020**

(52) **U.S. Cl.**

CPC ***G06Q 30/0201*** (2013.01); ***G06F 16/1724*** (2019.01); ***G06F 16/128*** (2019.01); ***G06F 16/178*** (2019.01); ***G06F 16/113*** (2019.01)

(57) **ABSTRACT**

Improved systems and methods for database management using data normalization and defragmentation techniques are provided. At least one exchange processor in communication with an exchange computer system receives market data from the exchange computer system, processes the market information, and transmits the market data to a master processor. The master processor receives the market data, processes the data using at least one normalization process to generate normalized data including an intra-day file and an archival file, and stores the intra-day file and the

Breaking it Down

What're the projects behind the offerings?

Capture

`mstcaprecv`

Captures raw data from network using kernel bypass.

`mstcapwrite`

Writes raw data to raw capture files.

`mstcapmon`

Generates quality metrics about captured data.

`mstsplit`

Post-processes raw capture into client-facing capture offerings.

`replay`

Replays pcaps onto the network for functional testing and simulation.

Normalization

`mstnorm`

Transforms market data to various formats (CSV, JSON, Parquet, et cetera) with or without additional aggregate measures.

`mstanalyze`

Pluggable, general purpose processing pipeline.

MDX

Proprietary, distributed, time series database engine.

High Performance Query

REST- and WebSocket-accessible interface for querying market data built on top of MDX.

Breaking it Down

What're the projects behind the offerings?

Capture

`mstcaprecv`

Captures raw data from network using kernel bypass.

`mstcapwrite`

Writes raw data to raw capture files.

`mstcapmon`

Generates quality metrics about captured data.

`mstsplit`

Post-processes raw capture into client-facing capture offerings.

`replay`

Replays pcaps onto the network for functional testing and simulation.

C++17

Normalization

`mstnorm`

Transforms market data to various formats (CSV, JSON, Parquet, et cetera) with or without additional aggregate measures.

`mstanalyze`

Pluggable, general purpose processing pipeline.

MDX

Proprietary, distributed, time series database engine.

High Performance Query

REST- and WebSocket-accessible interface for querying market data built on top of MDX.

Breaking it Down

What're the projects behind the offerings?

Capture

`mstcaprecv`

Captures raw data from network using kernel bypass.

`mstcapwrite`

Writes raw data to raw capture files.

`mstcapmon`

Generates quality metrics about captured data.

`mstsplit`

Post-processes raw capture into client-facing capture offerings.

`replay`

Replays pcaps onto the network for functional testing and simulation.

C++17

Normalization

`mstnorm`

Transforms market data to various formats (CSV, JSON, Parquet, et cetera) with or without additional aggregate measures.

`mstanalyze`

Pluggable, general purpose processing pipeline.

C++20

MDX

Proprietary, distributed, time series database engine.

High Performance Query

REST- and WebSocket-accessible interface for querying market data built on top of MDX.

C++20

LSEG

Breaking it Down

What're the projects behind the offerings?

Capture

`mstcaprecv`

Captures raw data from network using kernel bypass.

`mstcapwrite`

Writes raw data to raw capture files.

`mstcapmon`

Generates quality metrics about captured data.

`mstsplit`

Post-processes raw capture into client-facing capture offerings.

`replay`

Replays pcaps onto the network for functional testing and simulation.

C++17

Normalization

`mstnorm`

Transforms market data to various formats (CSV, JSON, Parquet, et cetera) with or without additional aggregate measures.

`mstanalyze`

Pluggable, general purpose processing pipeline.

C++23

C++20

MDX

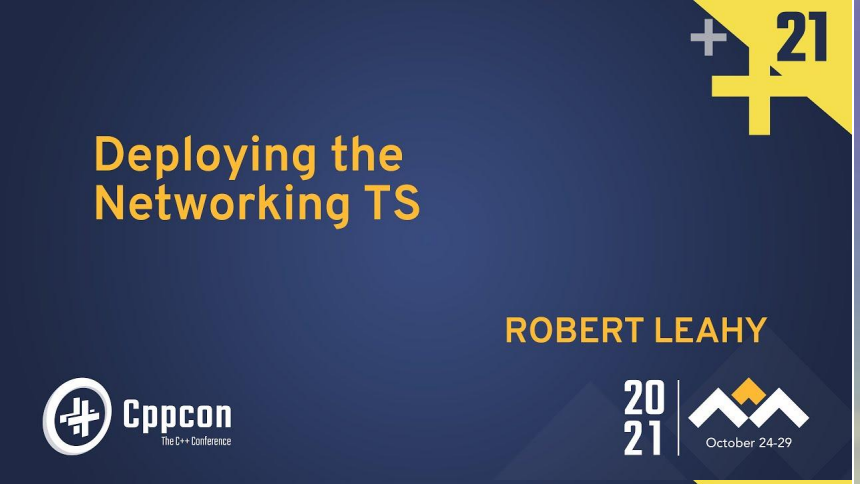
Proprietary, distributed, time series database engine.

High Performance Query

REST- and WebSocket-accessible interface for querying market data built on top of MDX.

C++20

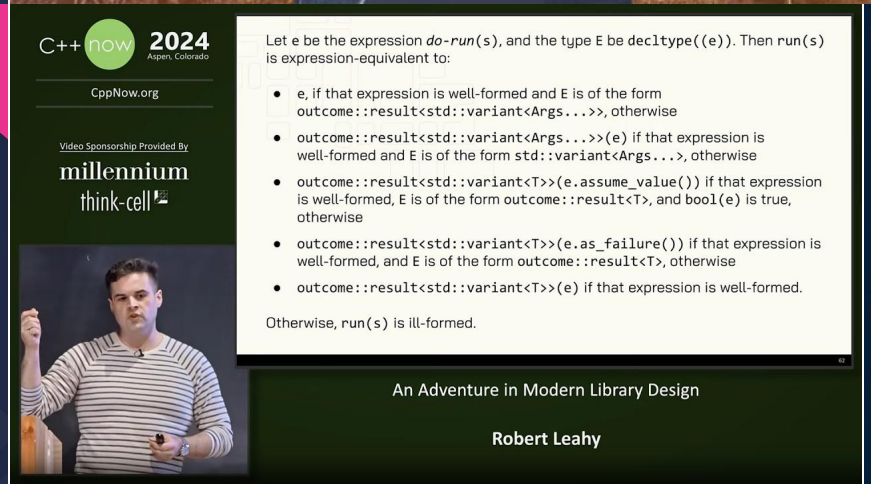
LSEG

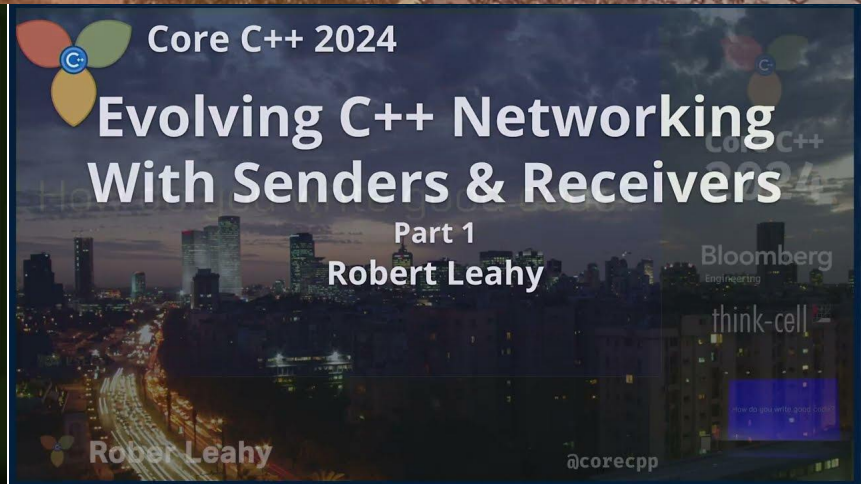

+ 21
Deploying the Networking TS
ROBERT LEAHY
Cppcon
The C++ Conference
2021
October 24-29


+ 21
Failing Successfully: Reporting and Handling Errors
ROBERT LEAHY
Cppcon
The C++ Conference
2021
October 24-29


+ 22
Taking a Byte Out of C++
Avoiding Punning by Starting Lifetimes
ROBERT LEAHY
Cppcon
The C++ Conference
2022
September 12th-16th


Fantastic Bugs and How to Test Them
Robert Leahy
2024


C++ now 2024
Aspen, Colorado
CppNow.org
Video Sponsorship Provided By
millennium
think-cell
Let e be the expression do-run(s), and the type E be decltype((e)). Then run(s) is expression-equivalent to:
e, if that expression is well-formed and E is of the form outcome::result<std::variant<Args...>>, otherwise
outcome::result<std::variant<Args...>>(e) if that expression is well-formed and E is of the form std::variant<Args...>, otherwise
outcome::result<std::variant<T>>(e.assume_value()) if that expression is well-formed, E is of the form outcome::result<T>, and bool(e) is true, otherwise
outcome::result<std::variant<T>>(e.as_failure()) if that expression is well-formed, and E is of the form outcome::result<T>, otherwise
outcome::result<std::variant<T>>(e) if that expression is well-formed.
Otherwise, run(s) is ill-formed.
An Adventure in Modern Library Design
Robert Leahy


Core C++ 2024
Evolving C++ Networking With Senders & Receivers
Part 1
Robert Leahy
Bloomberg
think-cell
Robert Leahy
@corecpp

Come to my lightning talk!

LSEG


HOTEL LOBBY
Nathan Owen

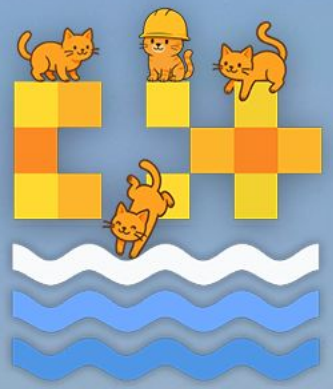
When Do You Know `connect` Doesn't Throw?

Document Number: P3388R2
Date: 2025-04-01
Reply-to: Robert Leahy <rleahy@rleahy.ca>
Audience: LWG

Abstract

This paper proposes a change which will enable earlier determination that connecting a certain sender with a certain receiver will never throw an exception.

Background



Extending `std::execution` Further

Higher-Order Senders and the Shape of Asynchronous Programs

Robert Leahy

2025

A Sentinel for Null-Terminated Strings

Document #:	P3705R0
Date:	2025-05-17
Project:	Programming Language C++
Audience:	SG-9 Ranges LEWG

Reply-to:	Eddie Nolan <eddiejnolan@gmail.com>

Contents

`std::constant_wrapper`

Document #:	P2781R8
Date:	2025-03-15
Project:	Programming Language C++
Audience:	LEWG LWG

Reply-to:	Hana Dusíková <hanicka@hanicka.net> Matthias Kretz <m.kretz@gsi.de> Zach Laine <whatwasthataddress@gmail.com>

Contents

Summary

What do we do?

LSEG Data & Analytics...

- ...writes modern C++
- ...improves C++ itself
- ...creates software for modern financial markets
- ...creates and supports services for modern financial market participants

LSEG DATA & ANALYTICS

We’re Hiring!

Barry Rindenow – Talent Acquisition Partner
barry.rindenow@lseg.com

https://www.lseg.com/en/careers

LSEG