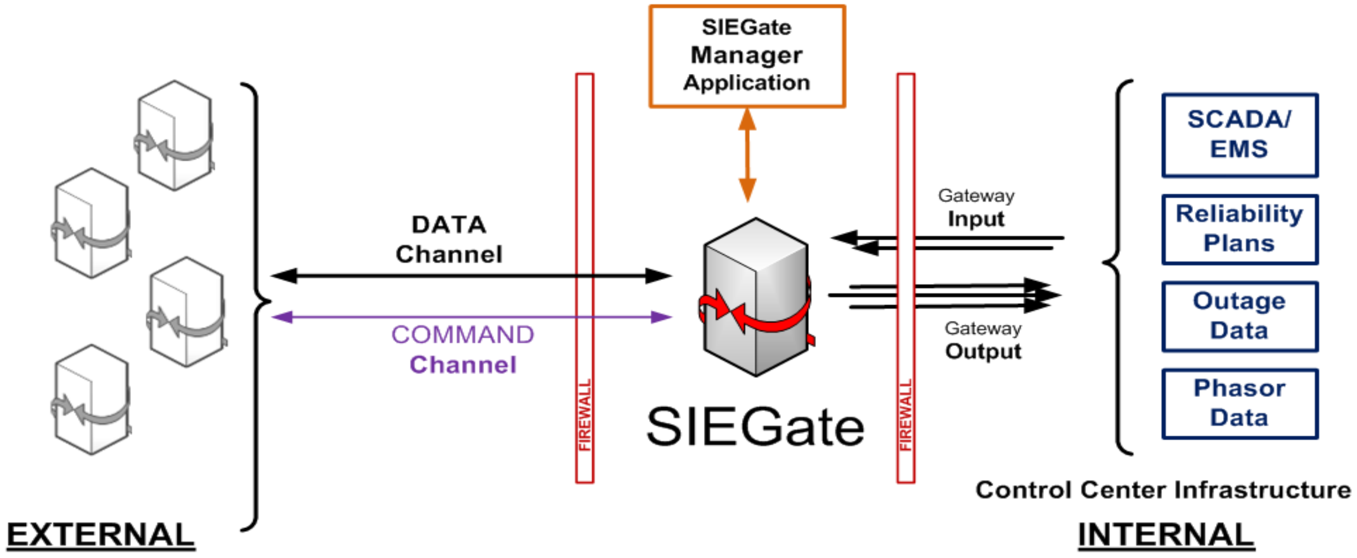
SIEGate Introduction: Overview and Core Architecture

Erich Heine & Tim Yardley



SIEGATE OVERVIEW

SIEGate: Summary

- **Objective**

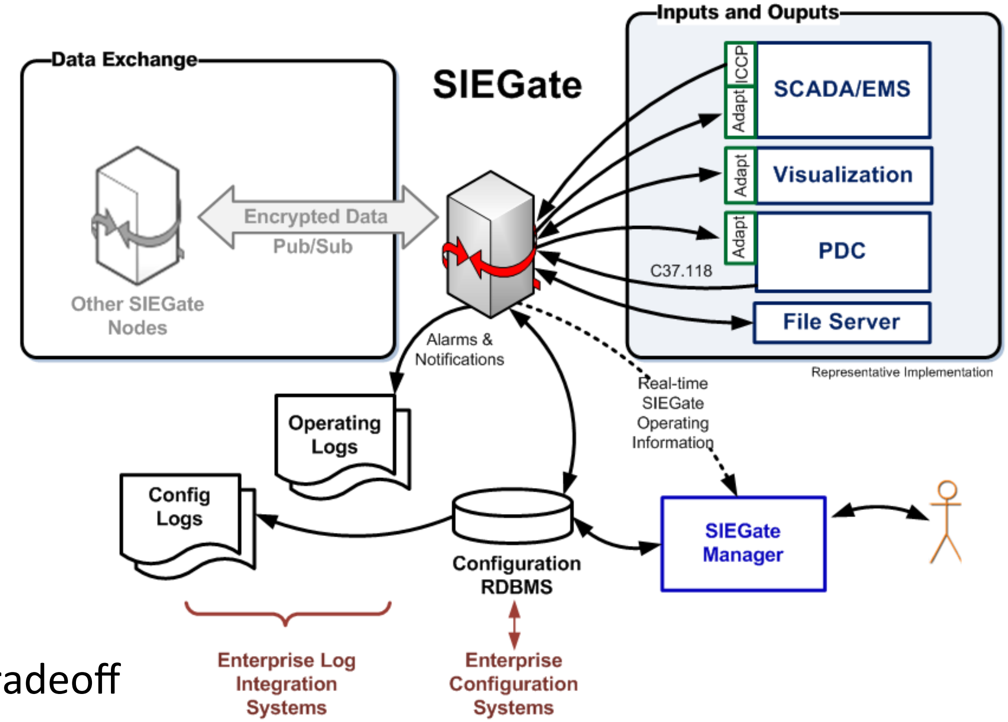
  To commercialize an appliance that enables the secure exchange of all types of reliability and market data among grid operating entities and provide a next-generation platform for GPA Open* products

- **Design Approach**
  - Lower risk by building upon the open source phasor gateway
  - Create an extensible platform
  - Design security throughout
  - Balance real-time and security needs
  - Conduct thorough bench tests to identify and fix security defects

- **Technical Goals:**
  - Maintain Time-series framework compatibility
  - Increase performance with new core
  - Separate low-level networking concerns from data handling concerns
- **Development Partners:** Grid Protection Alliance; University of Illinois
- **Test and Demonstration Partners:** Pacific Northwest National Laboratory, Alstom Grid, and PJM Interconnection

SIEGate: Technical Design Challenges

- Performance given system complexity
  - Support multiple data types efficiently and securely
  - Support multiple priorities
  - Minimize latency and maximize throughput
- High availability assurance
  - Horizontal and vertical scalability
  - SIEGate stability and reliability
  - Graceful performance degradation
- Security assurance
  - Maximize security performance
  - Minimize security breach impact
  - Configurable security levels
  - Security versus simplicity/usability tradeoff



SIEGate: Technical Design Principles

- Minimize thread-locking and contention
- Pre-compute criteria for decisions rather than on-the-fly
- Simplify resource access
- Choose heavy memory usage over heavy CPU
- Discard unneeded data as early as possible
- Provide extensibility & offloading
- Adhere to the single responsibility principle
- Maintain a layered approach to security and defenses
- Design components to operate with least privilege
- Leverage existing, tested components
- Pluggable component architecture

SIEGate: Data Inputs and Outputs

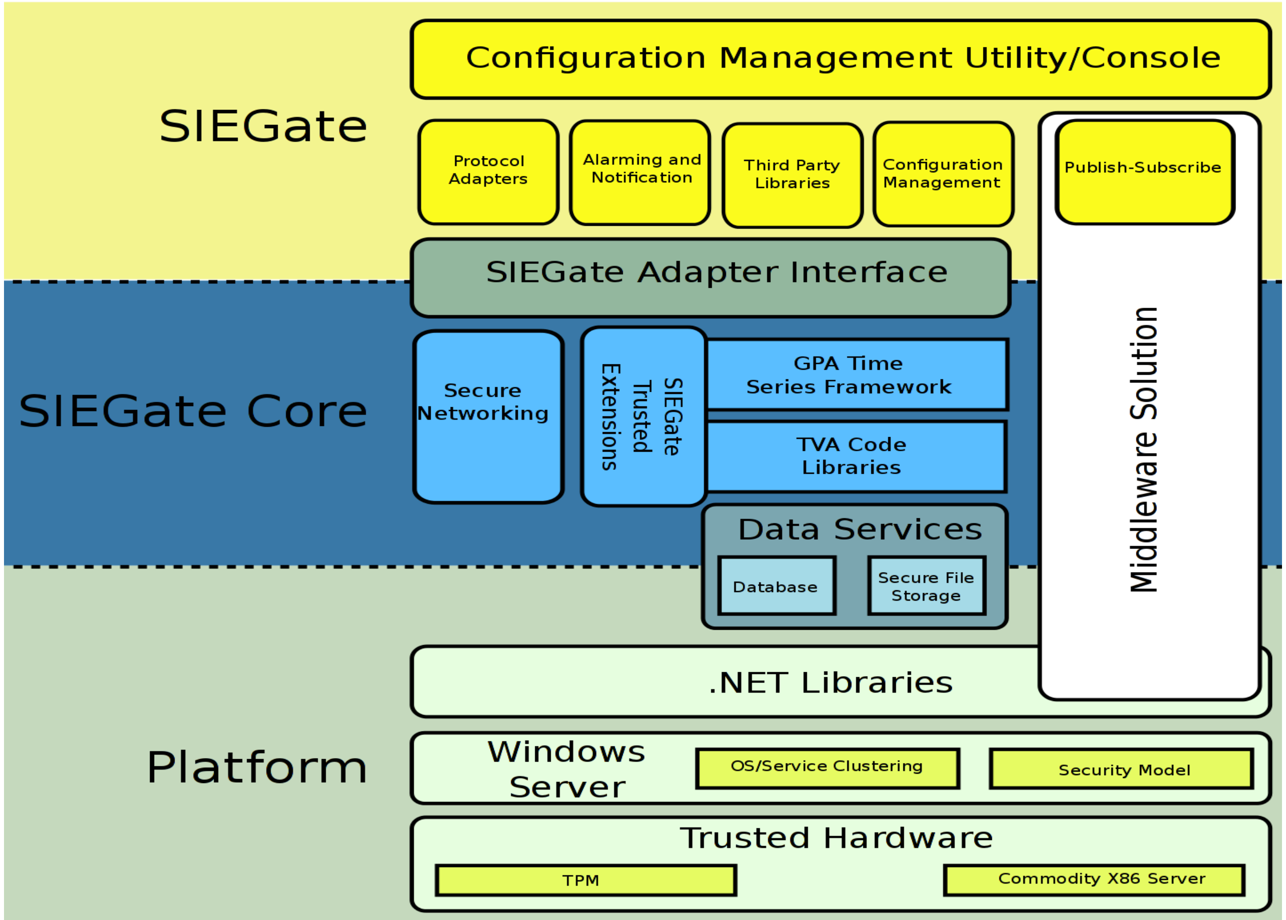
	Input	Output	Decomposition
• **SCADA Protocols**			
– DNP3, ICCP	Yes	No	Yes
• **Phasor Protocols** – e.g., 61850-90-5, C37.118	Yes	Yes*	Yes
• **File Based Data** – e.g., SDX, COMTRADE, PQDIF	Yes	Yes	No
• **Exchange Protocols and APIs** – SIEGate Exchange Protocol (SGEP)	Yes	Yes	

<u>Deferred</u>

* Stream Mirroring Allowed

	Input	Output	Decomposition
• **Process Control Protocols** – 61850, OPC – DNP3, ICCP Control and Output	Yes	No	Yes

SIEGate Component Architecture

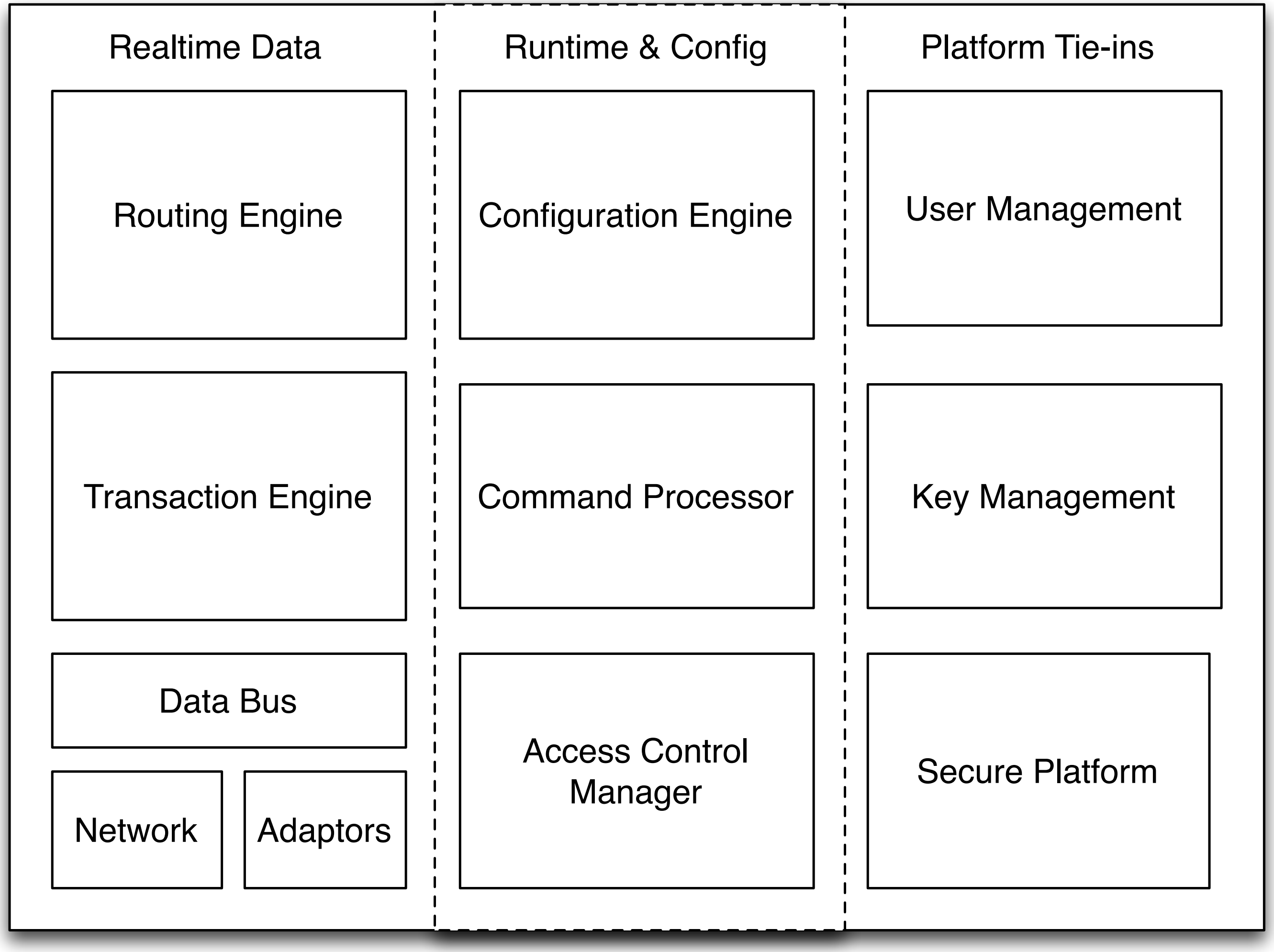


Application and Platform

- Platform
  - Windows server core
  - Take advantage of .NET 4.5 features and improvements
  - Take advantage of hardware security and performance capabilities
- Application
  - Maintain focus on the Input, Action and Output Adaptor paradigm
  - Ease the use of 3rd party components
  - Improved handling of alarms and notifications
  - Downstream developer continuity

SIEGATE CORE

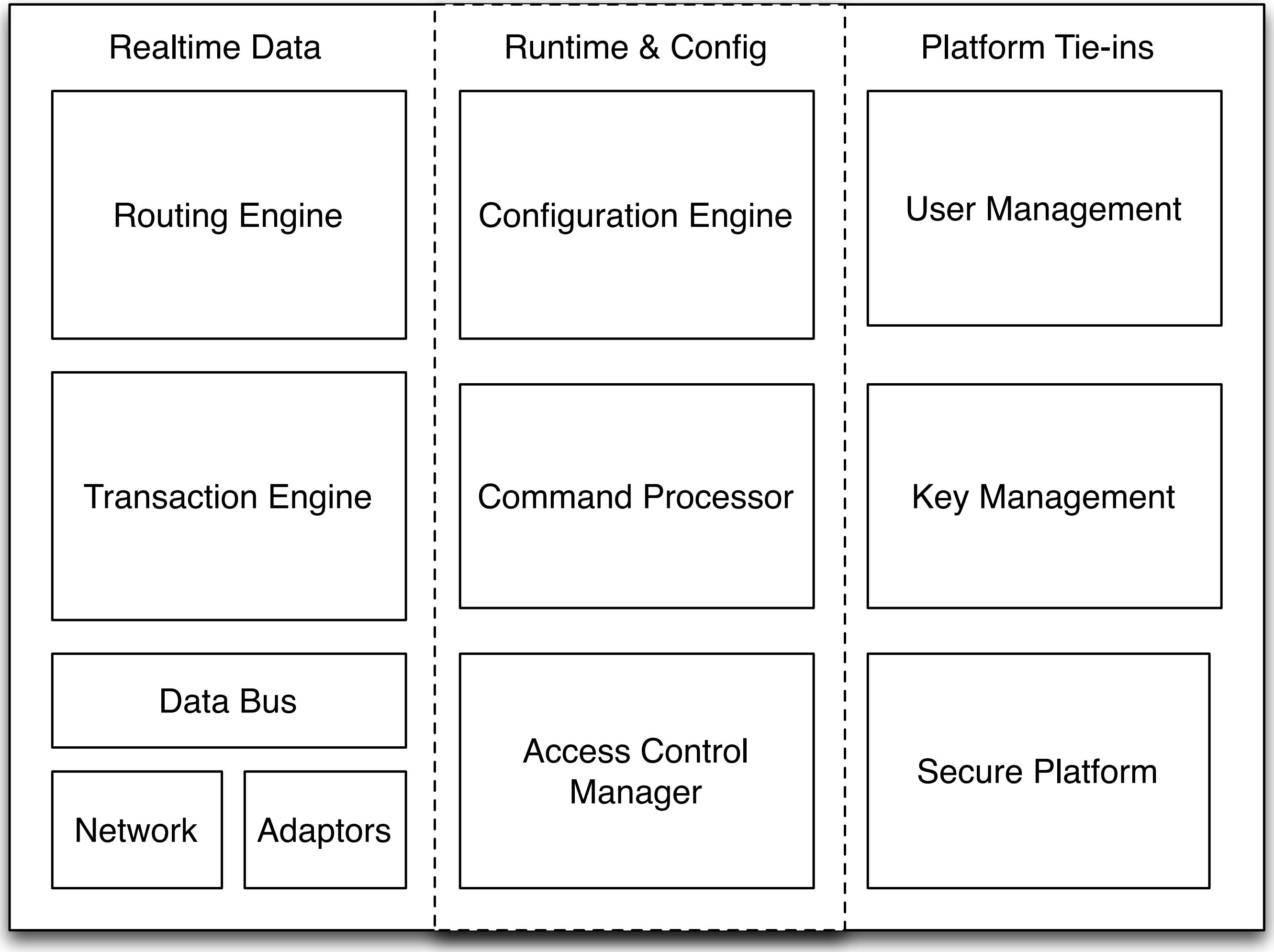
SIEGate Core Overview



Real-time Data

- Routing Engine
  - Arranges adaptors on the data bus
  - Configures data-path for QoS
- Transaction Engine
  - Marks and tracks data for receipt guarantee
  - Handles re-transmission QoS concerns related to this
- Network, Data Bus and Adaptors
  - Main data-path through system
  - Lightweight and speed oriented
- This Silo will be revisited in the data-path section

SIEGate Core Overview



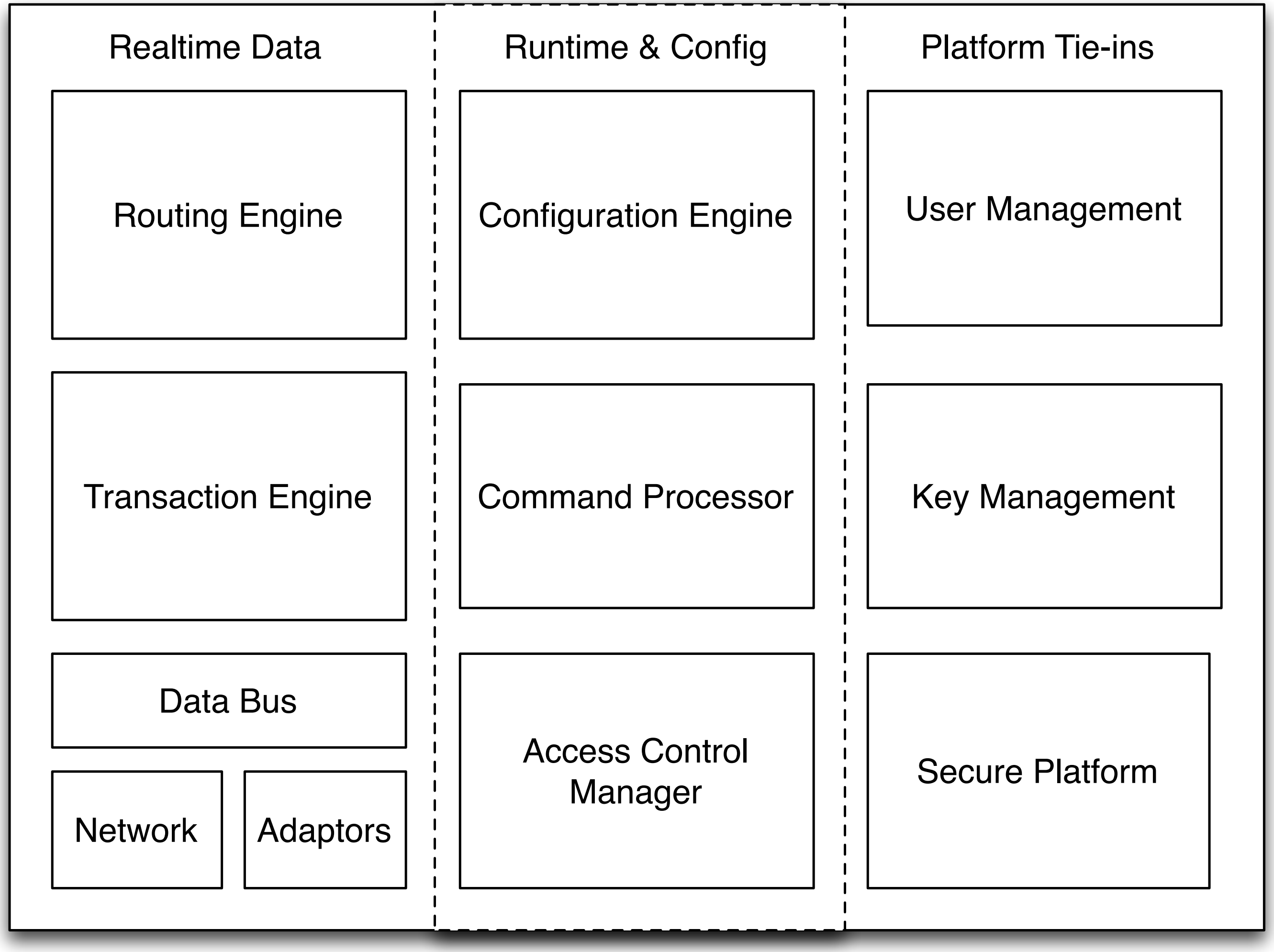
Runtime & Config

- Configuration engine
  - Translates stored configuration console information for the routing engine and data-path
  - Handles configuration change events
  - Initializes and/or updates data-path objects on configuration changes
  - Provides hook points for commands from the Management Console

Runtime & Config pt 2

- Command Processor
  - Coordination with remote SIEGates
    - Key Changes
    - Failover
  - Measurement stream subscription
    - Handles subscription requests
    - Ensures permissions are adequate (via ACLs)
- Access Control manager
  - Tracks publish and subscribe permissions between SIEGates
  - Only allows connections from configured internal devices
  - Leverages user management for console operations

SIEGate Core Overview



Platform Tie-ins

- User Management
  - Uses Windows/AD users to enforce roles
- Key Management
  - No 3rd parties necessary
  - Tracks and stores keys from trusted partners for SIEGate subscription (in conjunction with ACLs)
- Secure Platform
  - Hooks for secure logging
  - TPM support
  - .NET security features

CROSS CUTTING COMPONENTS

Trusted Union

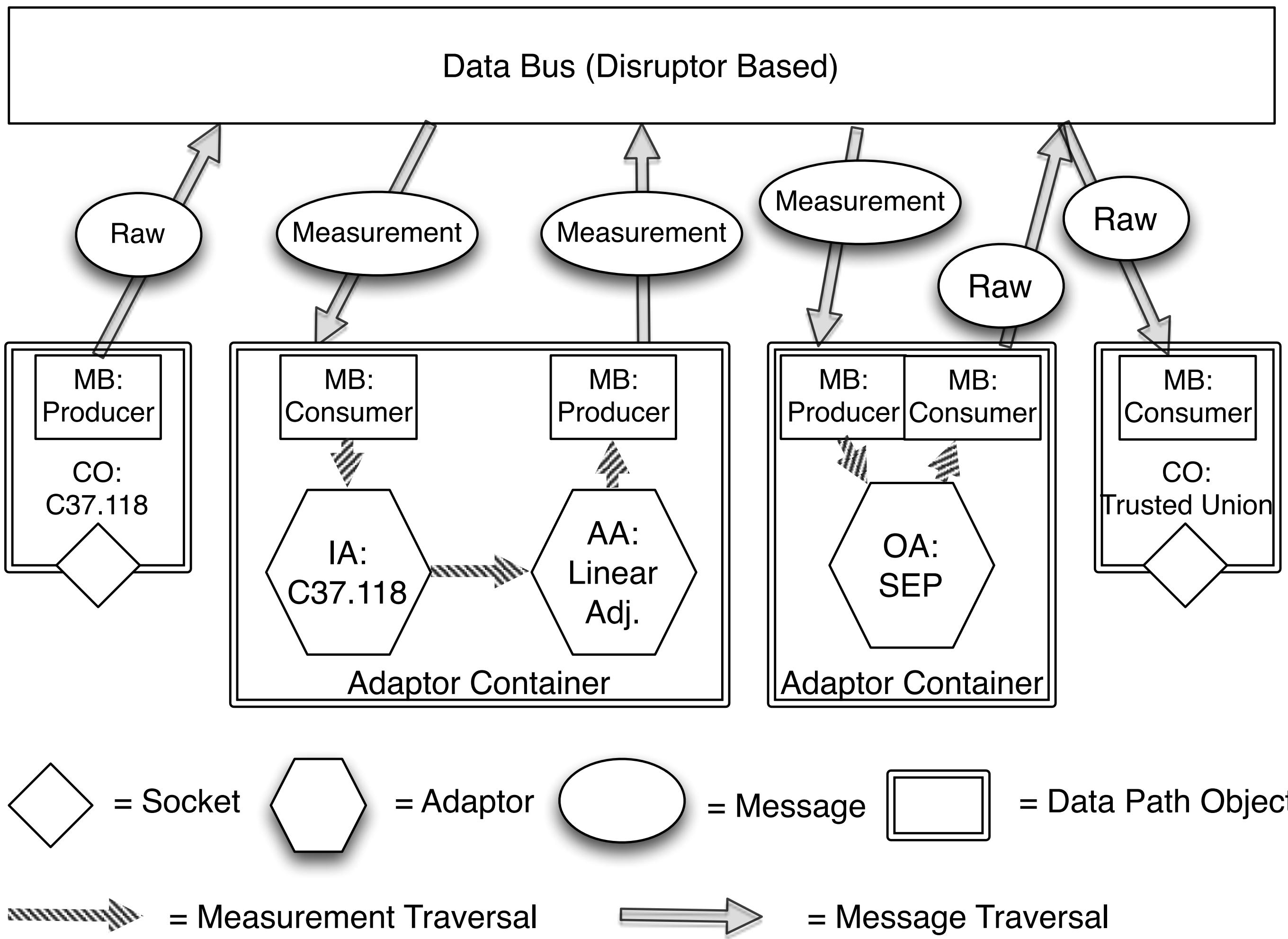
- Secure link between SIEGates
  - Measurement based transport protocol
  - TLS encryption
  - Receipt confirmation for important data
- Separate Command channel
  - Handles SIEGate coordination
  - Handles measurement subscription and publishing
  - Extensible for failover patterns, quality of service changes on-the-fly and so on in future versions

Leveraging .NET

- .NET Events based Event System
  - Asynchronous and lock free internal commands
  - Allows looser coupling of components
- Secure logging
  - Allows better security monitoring
  - Improved accountability
  - Statistical information

DATA-PATH IN DEPTH

Data path



CO: Communication Object   MB: Message Bus
IA: Input Adaptor OA: Output Adaptor AA: Action Adaptor

Adaptor Paradigm

- Splits data-path into 2 parts
  - Measurement path
  - Message path
- Maintains the concept of Input, Output and Action adaptors, in “pure” form
- Provides better adherence to Single Responsibility Principle
- Strong “measurement” abstraction – internal core complexity separated from adaptors

Practical effects

- Measurements are immutable
  - Adaptors that change values must output new StreamIDs
  - More measurement types
- Input Adaptors are now 2 parts
  - Network Communication object
  - Input Adaptor becomes data parser
- Similar effect for Output Adaptors

Measurements, Files and Messages

Immutable Objects

- Measurement:
  - Stream ID
  - Timestamp
  - Quality
  - Data
- File:
  - Stream ID
  - Timestamp
  - Quality
  - Data Buffer Pointer
- RawData, Frame

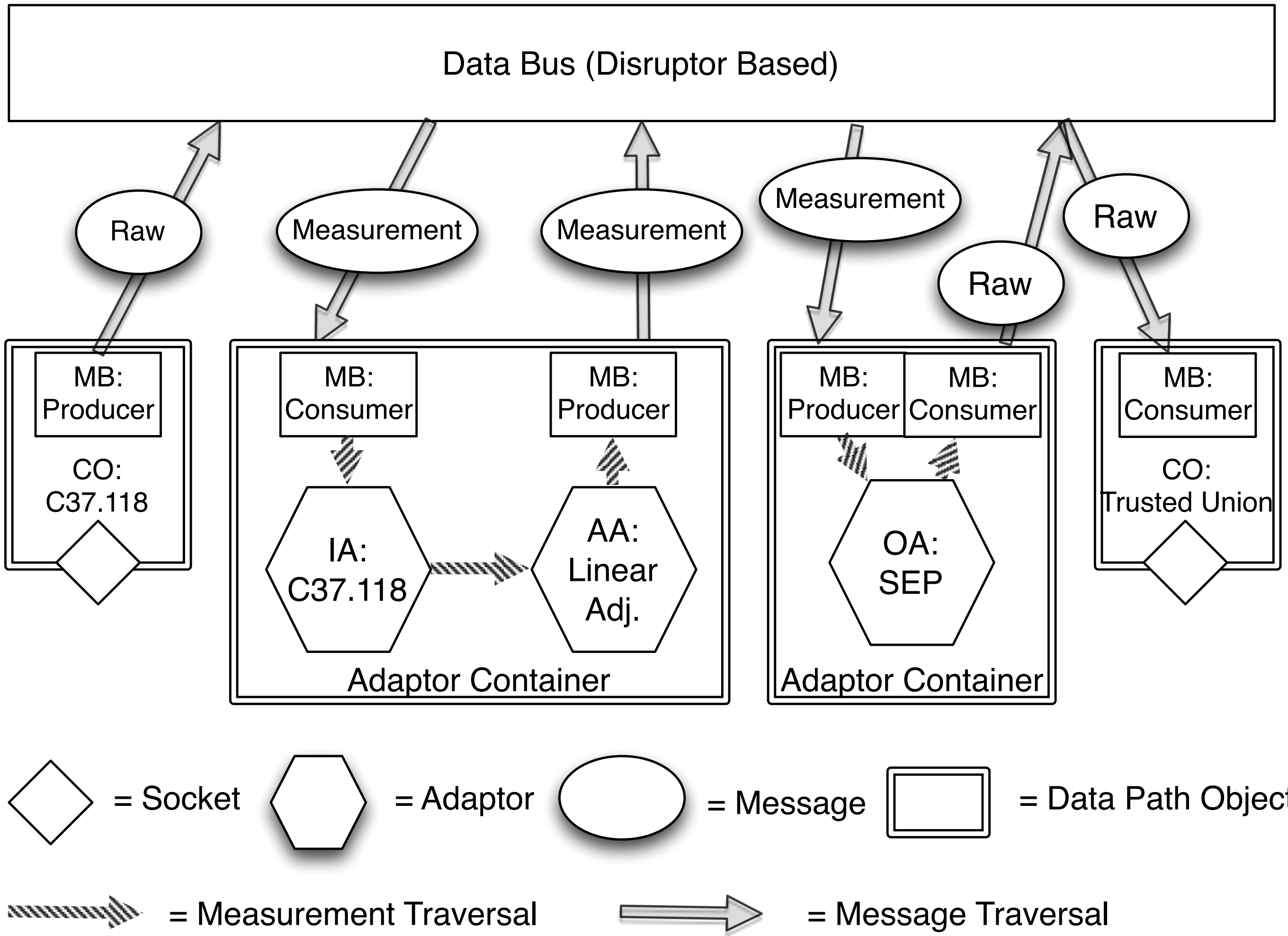
Mutable Object

- Message:
  - Stream ID
  - QoS Parameters
  - Transaction Flags
  - Other System info?
  - Message/File Object

Benefits – Why the fuss?

- Enables idempotent adaptors
  - Enhances optimization options
  - Provides better filtering capabilities
  - Reduces internal overhead in certain scenarios
- Reduced locking overhead
- No need for memcopy in the data path
- Sandboxing dangerous bits
  - Unstable adaptors
  - Network communications

Data path



CO: Communication Object MB: Message Bus
IA: Input Adaptor OA: Output Adaptor AA: Action Adaptor

Below the adaptors

- Internal message passing provided by Disruptor
- Protocol Communication objects to handle socket communications
- Quality of Service by thread scheduling and careful composition of Adaptor Containers
- Configuration and chaining handled by routing engine

Current status

- Skeleton of the core complete, based on Disruptor
- Optimizations to GUID stream identifier mechanisms
- Adaptor base classes in place
- Currently coding:
  – Command processor
  – Access control and permissions
  – Tests of core architecture

Questions?