



NMI-EDIT End-to-End Diagnostic Backplane

Internet2 Middleware End-To-End Diagnostic
Advisory Group

Performance Measurement Architecture Workshop
San Diego Supercomputer Center (12/11/03)
Chas DiFatta (chas@cmu.edu)

- Problem Space
- Architecture
 - Collection
 - Event Record
 - Management Backplane
 - Application API
- Applications
- Status

Banes of the Distributed System Diagnostician

- No **access** to the diagnostic data
- **Discovering** valuable information in a sea of data
- **Correlating** different diagnostic data types
- Providing evidence to prove or **repudiate** a diagnosis
- Finding **time** to create tools to transfer diagnostic knowledge to less skilled organizations and/or individuals

What if...

- The collection and management of a wide variety of event data was made simple
- There was a simple yet highly configurable dissemination infrastructure
- All application, host and network events could be correlated with each other
- There existed an API where developers could rapidly build new tools that use this rich set of data and its management infrastructure

Goals of Effort

- **Assists the deployment efforts** in the configuration and testing phases
- **Reduces the problem discovery and analysis time** during the operation and maintenance phases

Goals of Effort Cont.

Assists application, developers, administrators and operators by;

- **Allowing the collection** of a wide variety of diagnostic log formats
- **Establishing common event records** to share inter and intra-organizationally, to aid in the proof and repudiation of faults
- **Providing access** to application, system and network events that are both internal and external to the application to aid the bug fixing process
- **Enabling rapid development of diagnostic applications** for pre and post production phases

Outline

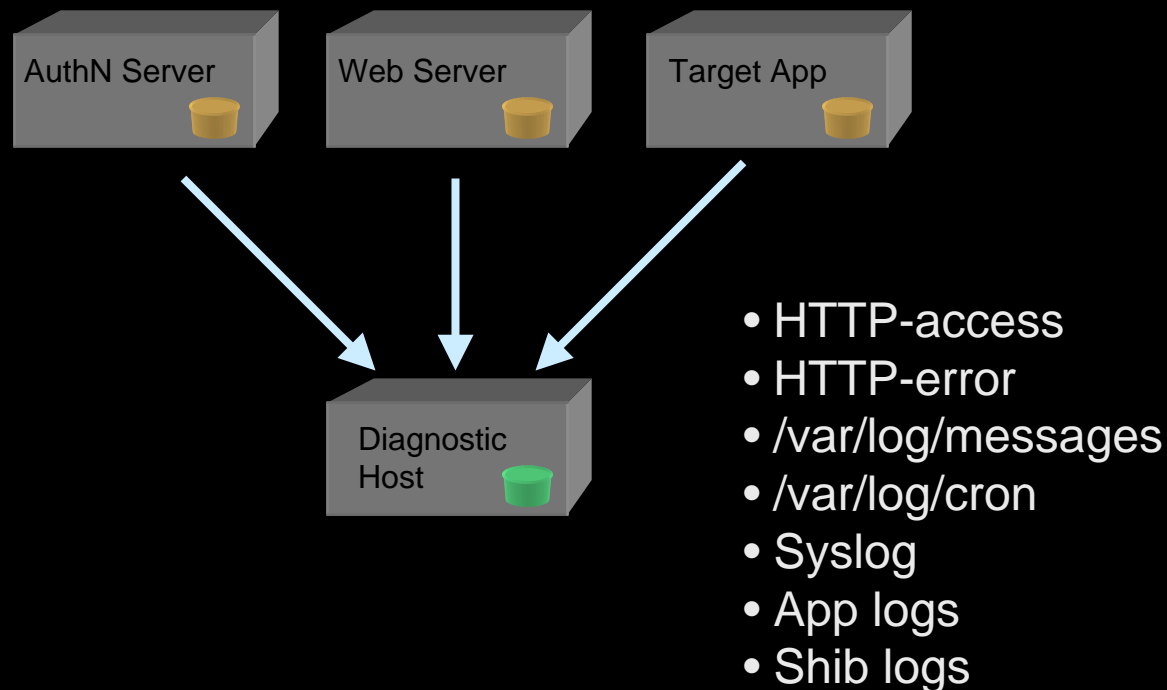
- Problem Space
- **Architecture**
 - Collection
 - Event Record
 - Management Backplane
 - Application API
- Applications
- Status

Goals of Architecture

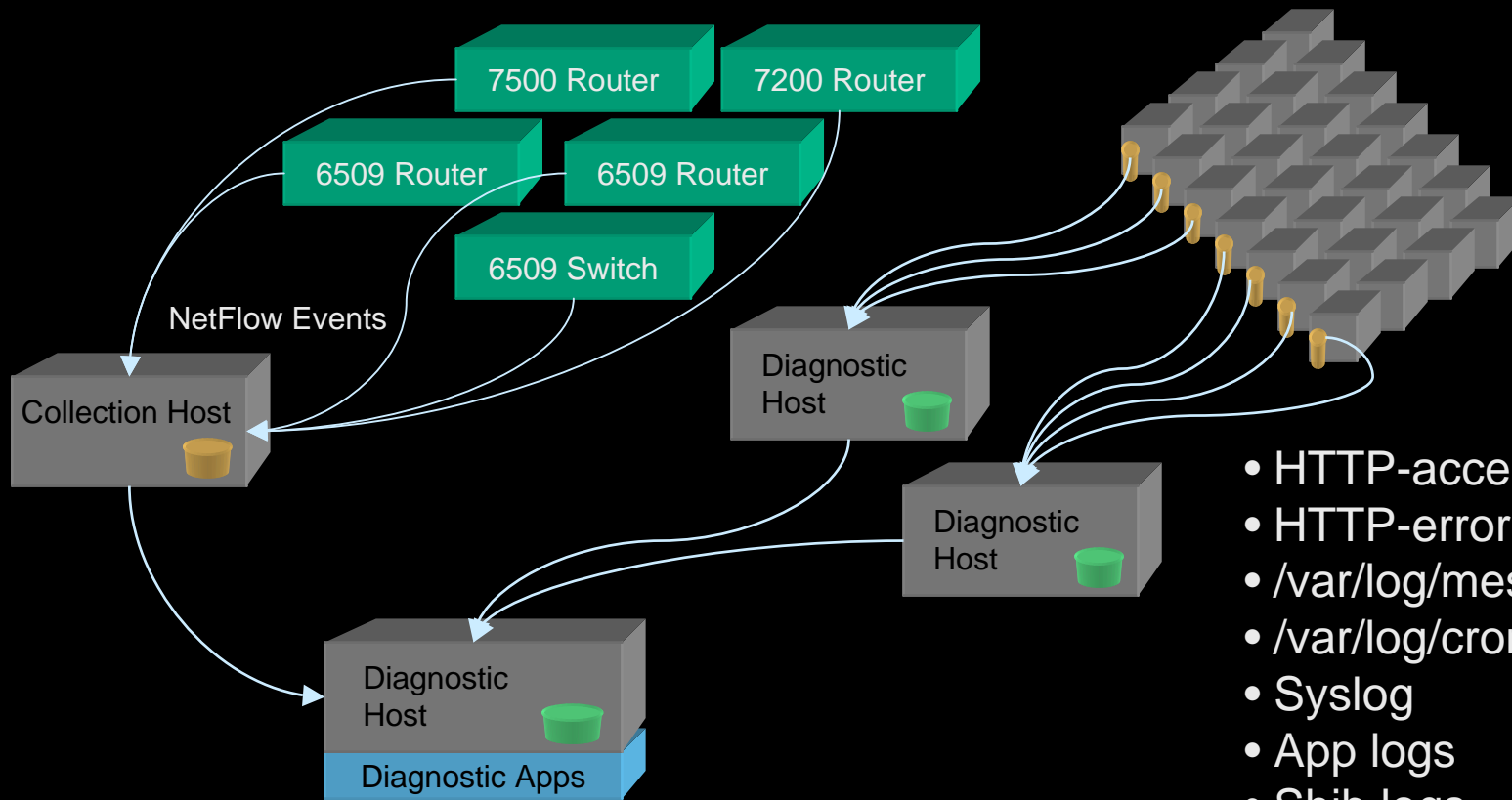
- **Expose** a wide array of diagnostic data to diagnostic tool developers
- Make it simple to **add new data types** without any modification to logging process
- Create a dissemination facility that is **scalable, secure and reliable**
- **Provide atomic methods** to filter, aggregate, classify, manipulate, search, and anonymize the data
- Create an simple **API** to use the methods
- **Deploy** the methods on popular development platforms

Simple Implementation

Enterprise Web Enabled Application



Enterprise Event Collection

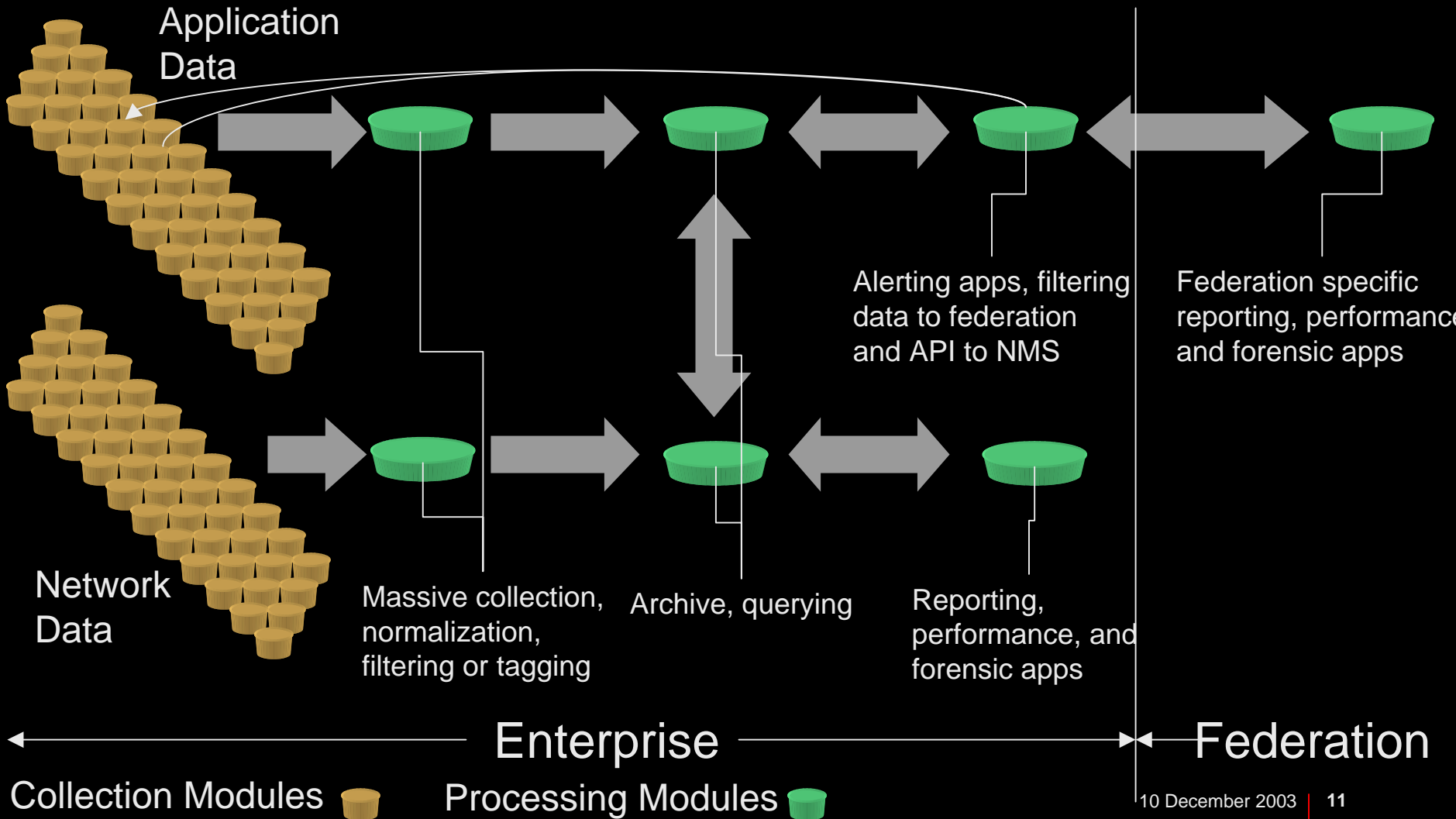


- HTTP-access
- HTTP-error
- /var/log/messages
- /var/log/cron
- Syslog
- App logs
- Shib logs
- NetFlow

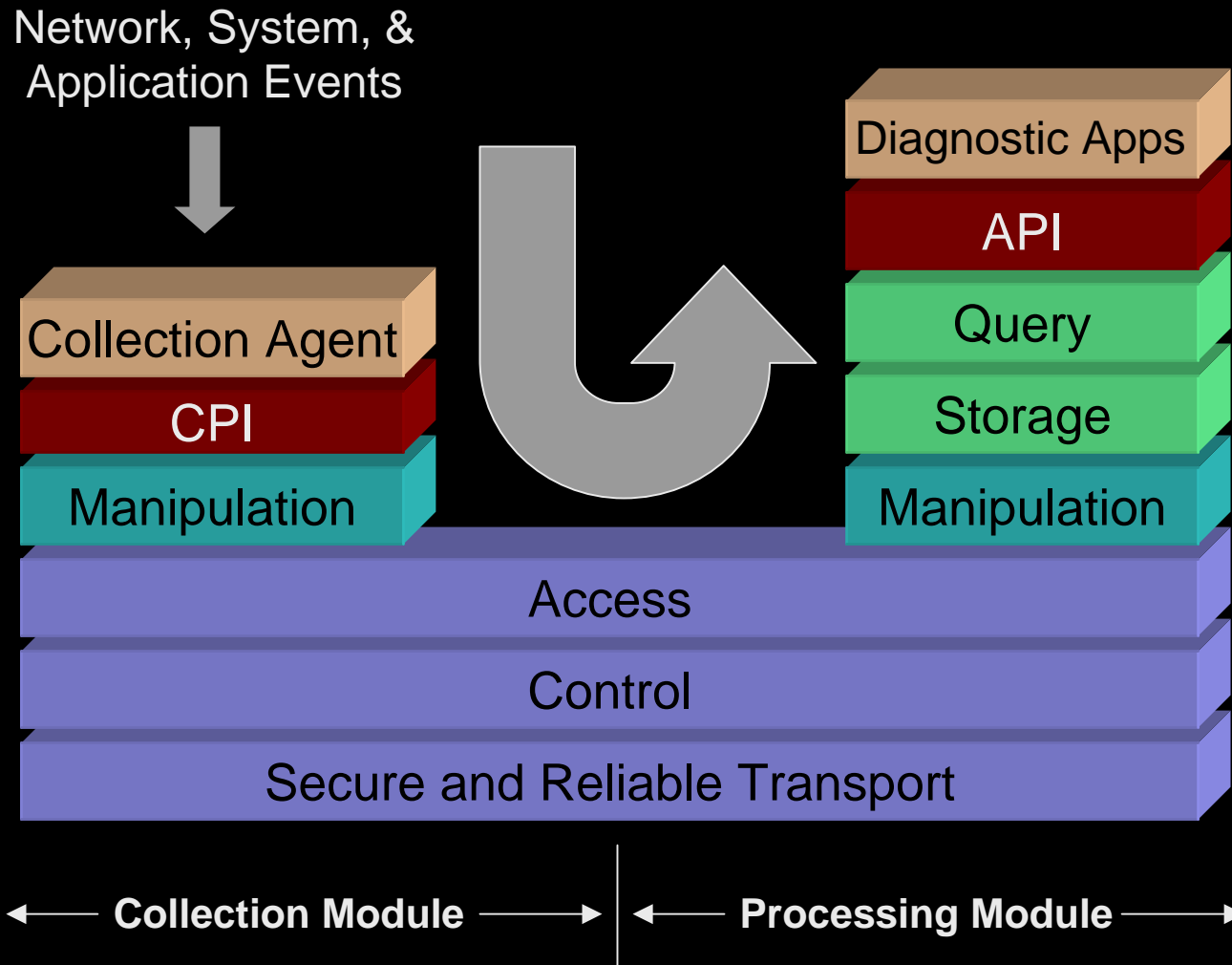
Collection Module

Processing Module

Complex Implementation



Backplane Elements



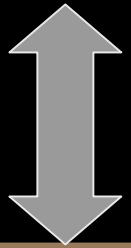
- Problem Space
- Architecture
 - Collection of Events
 - Event Record
 - Management Backplane
 - Application API
- Applications
- Status

What types of input events?

- Application and Host
 - Syslog
 - /var/log/*
 - http-access/error
 - MS specific (security, application and event logs)
 - Application specific (AFS, SHIB, LDAP, SMTP, etc.)
 - “many others”
- Network
 - NetFlow
 - SNMP
 - RMON
 - “others?”

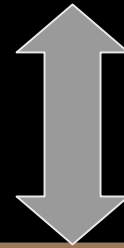
Collection Module

To Processing Modules

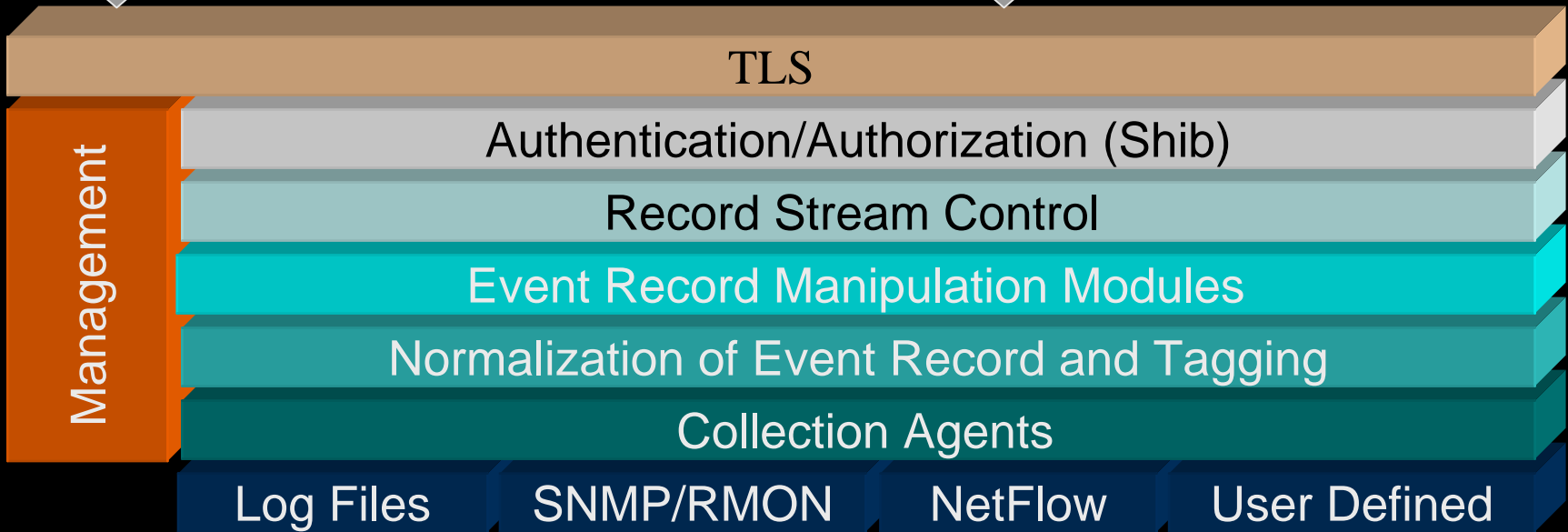


- Management of Internal Events
- Module Control
- Configuration

To Processing Modules



- Observed External Events



Collection Modules

- Operators and be piped in any order
- All operators are optional except for Normalization
- Normalization operator is unique to each input stream

Operators

Filter

Anonymizer

Aggregation

Normalization (collection agent)

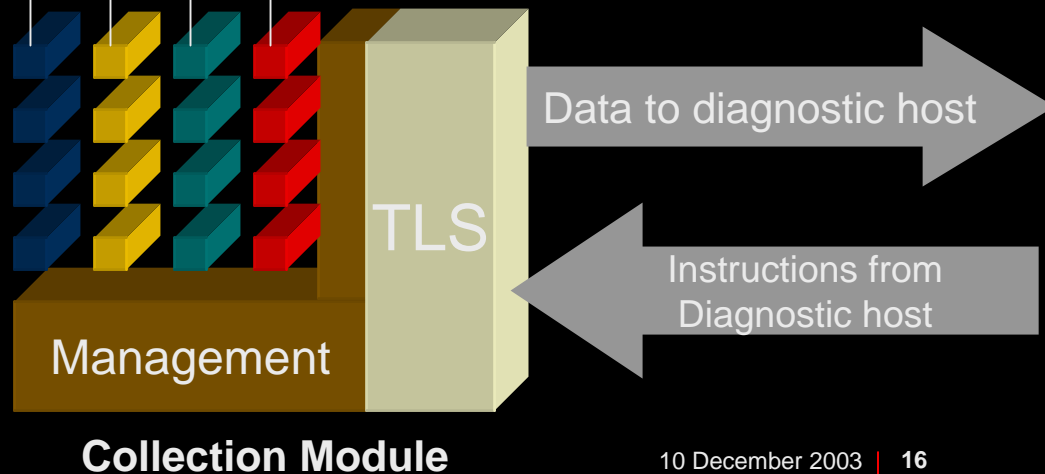
Input Stream Examples

App (Shibboleth) Log

Unix/WIN Log

Network (SNMP, NetFlow)

Http Log



Collection Module Configuration

Collection Modules

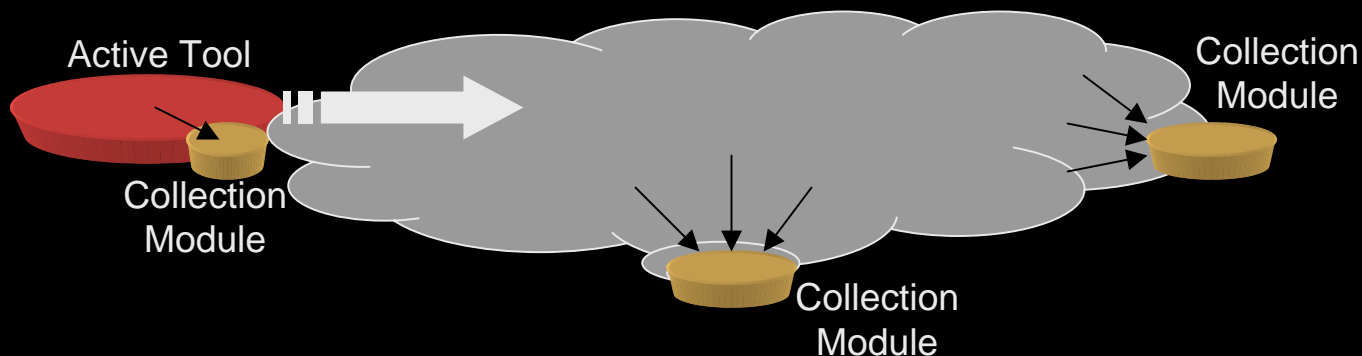
Processing Module



- Diagnostic domain definition (where do I get resources?)
- Module configuration (what do I become?)
- Download needed collection agents (what event do I observe?)
- Retrieve relevant event parsers (how do I decode/tag if needed?)

Passive and Active Measurement

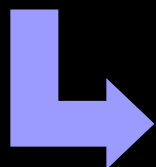
While 99% of event collection is passive, there is a large requirement for incorporating active mechanisms to enable performance and verification based tools



When active tool begins probing into the infrastructure, it calls a simple API that injects an event into the backplane, with a tagID="sometool" and Status="measurement"

- Problem Space
- Architecture
 - Collection of Events
 - **Event Record**
 - Management Backplane
 - Application API
- Applications
- Status

Event Descriptor Meta Field



- **Version**
- **Location** – IPAddr of collection agent, IPAddr of observation
- **ID** – unique identifier
- **Time** - start/stop
- **IP Address(es)** – source/(destination)
- **Source Class** – application, network, system, compound, bulk, management
- **Source Type** – file, stream, polled, interrupt
- **Event Name Tag** – (null), user defined (can be multiple tags)
- **Status** – normal, informational, warning, measurement, critical, error
- **Major Source Name** – filename, Netflow, Syslogd, SNMP, shell, etc.
- **Minor Source Name** – logging process name, SNMP variable name, etc.
- **Minor Source Version** - Apache/1.3.14, Bind/9.1, Netflow/1.0, SNMP V2, etc.
- **Encoding** – Binary, ASN1, ASCII, XML, etc.
- **Modified** – true/false

Event Record

Raw Event Element Tags



- **Tag Name** – User defined
- **Encoding** - Binary, ASN1, ASCII, XML, etc.
- **Value** – Assigned Value

·
·
·

- Indirection into the raw event data
- Data tags can be inserted at event record creation time or after
- Data tags retain their values if the raw event data is modified

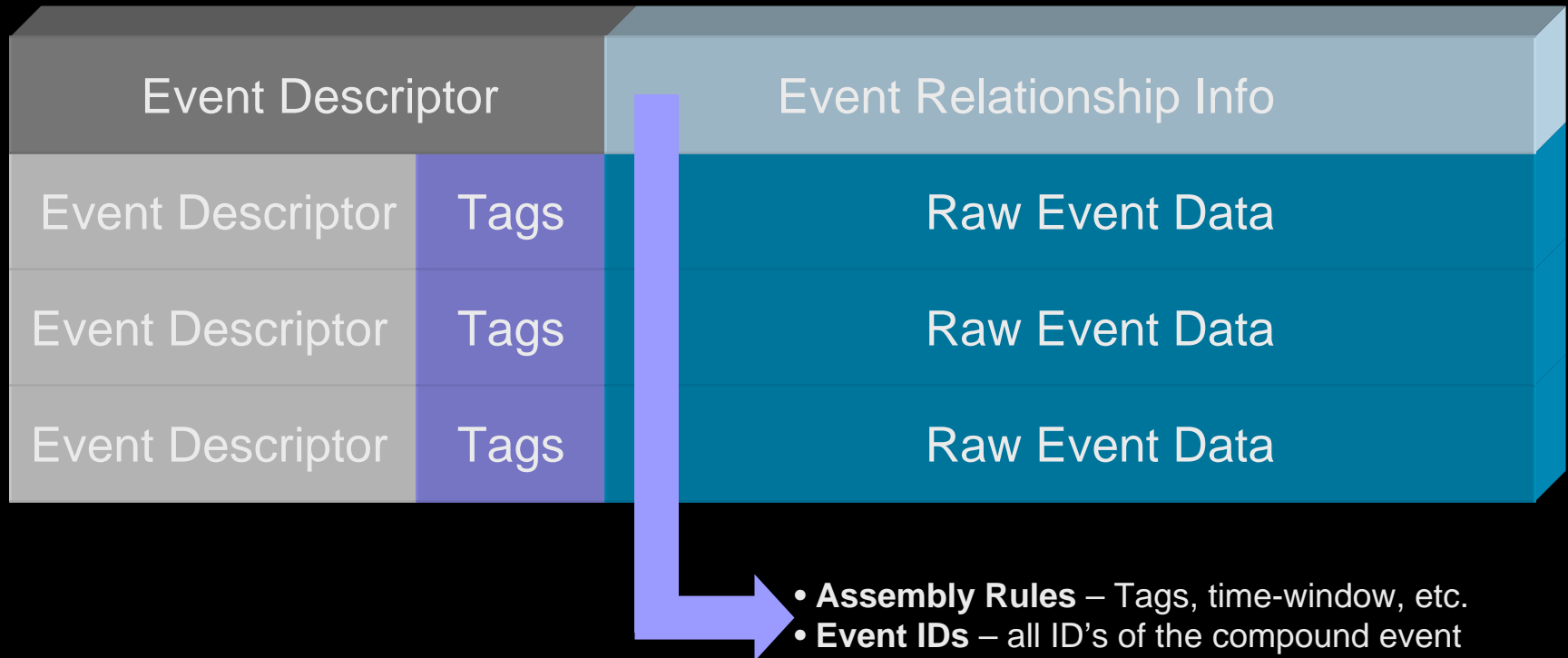
Compound Event Records

- Normalization of each specialized diagnostic data feed type (SHIB, IM, DIR, Syslog, P2P, etc.) into a common event record
- The tagging of specific events to help downstream correlation processes



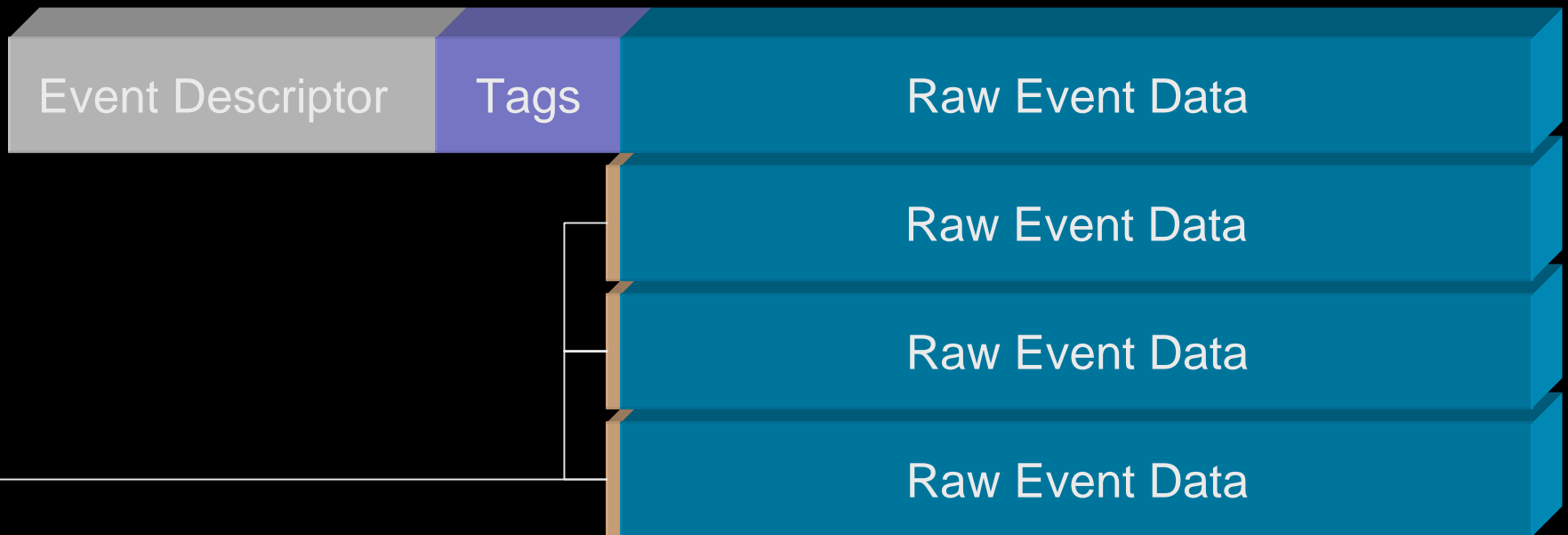
Event Record

Compound Event



Event Record

Bulk Events



- **Time** – Start/Stop
- **Status** - normal, informational, warning, measurement, critical, error

▪
▪
▪
▪

Internal Summary Event

Event Descriptor

- **Version**
- **ID** – unique identifier
- **Time** - start
- **IP Address** – source
- **Source Class** – management
- **Status** – informational
- **Module Name** – collection, processing

Module Summary Record

- Uptime
- Peripheral Process Statistics
 - management
 - archive
 - database
- Input stream sources
 - bytes, errors
- Output stream destinations
 - bytes, errors

Internal Summary Event

Event Descriptor

- **Version**
- **ID** – unique identifier
- **Time** - start
- **IP Address** – source
- **Source Class** – management
- **Status** – informational
- **Module Name** – collection, processing

Module Summary Record

- Uptime
- Peripheral Process Statistics
 - management
 - archive
 - database
- Input stream sources
 - bytes, errors
- Output stream destinations
 - bytes, errors

Internal Event

Event Descriptor

- **Version**
- **ID** – unique identifier
- **Time** - start
- **IP Address** – source
- **Source Class** – management
- **Status** – informational
- **Module Name** – collection, processing

Internal Event Details

- Stream X was dropped
- File not found
- Access denied
- Agent stopped
- Etc

Event Record Overhead

External Event

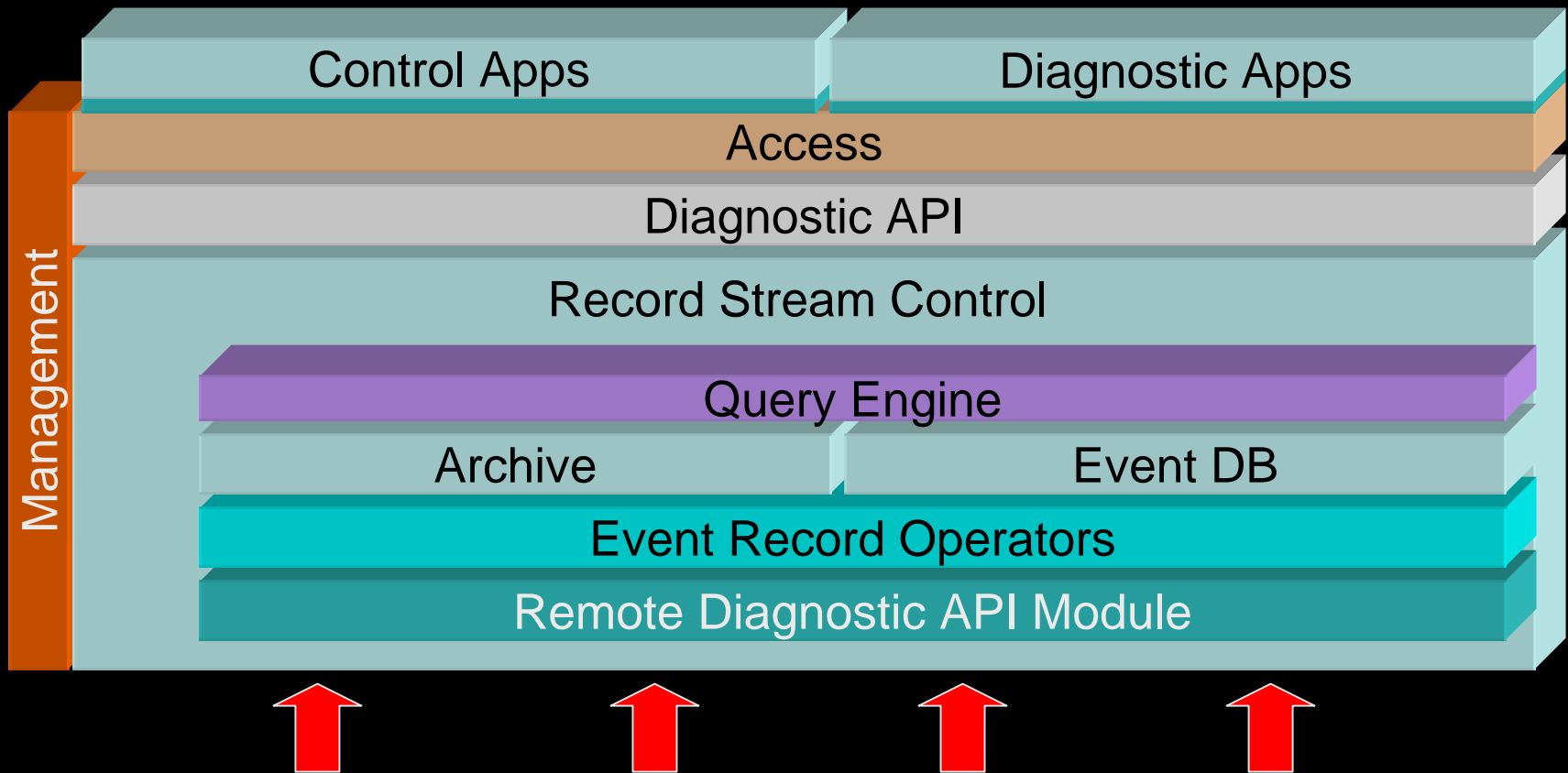


- **ID** – 12bytes
- **Time** – 24 or 12 bytes
- **IP Address(es)** – 8 or 16 bytes
- **Source Class** – 2 bytes
- **Source Type** – 2 bits
- **Event Tag** – 0 to 32 bytes typical (can be as large as 256)
- **Status** – 1 byte
- **Major Source Name** – 0 to 32 bytes typical (can be as large as 256)
- **Minor Source Name** – 0 to 16 bytes typical (can be as large as 256)
- **Minor Source Version** - 0 to 16 bytes typical (can be as large as 256)
- **Encoding** – 2 Bytes

- **Type** – External (observed), Internal (management)
- **Version**
- **Location** – IPaddr of collection agent, IPaddr of observation

- Problem Space
- Architecture
 - Collection of Events
 - Event Record
 - **Management Backplane**
 - Application API
- Applications
- Status

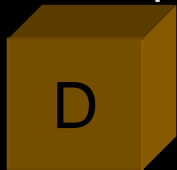
Processing Module Elements



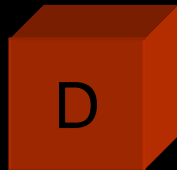
Internal Operators

Data Management/Access

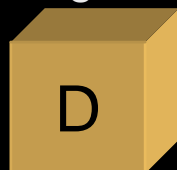
Internal Monitoring
Housekeeping



AuthN/
AuthZ

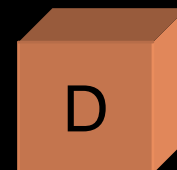


Control and
Configuration

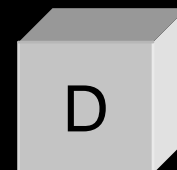


Data Repository

Archive

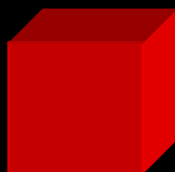


DB



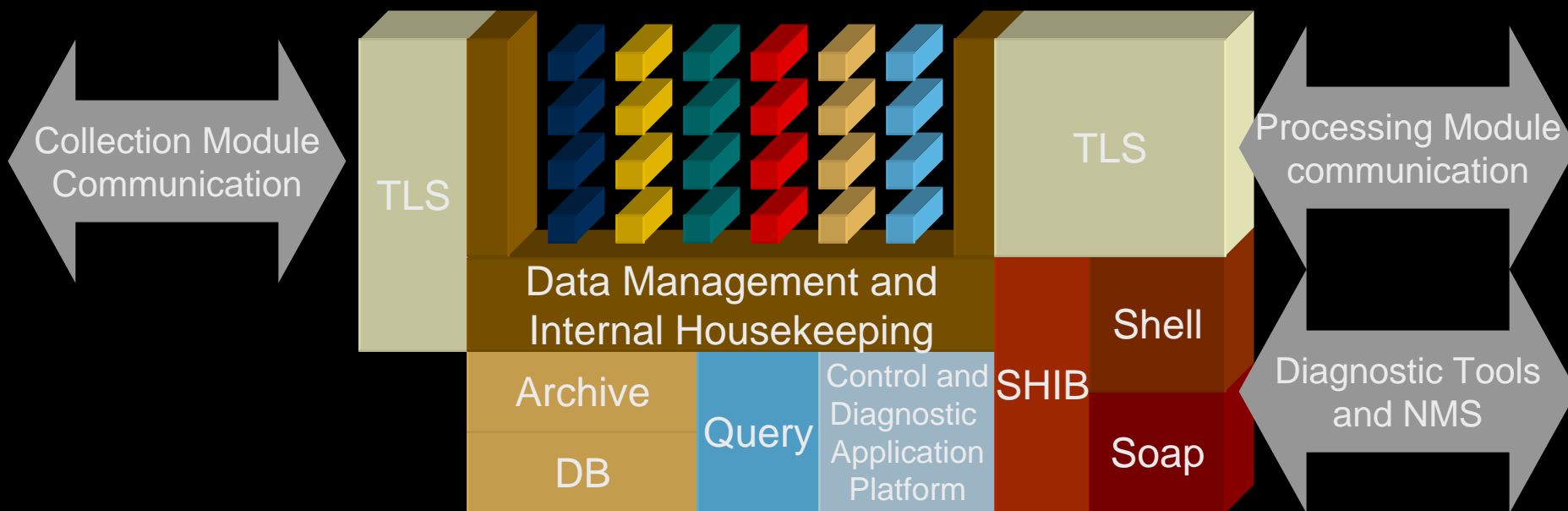
Record Manipulation

Filter Aggregator Anonymizer Tagging



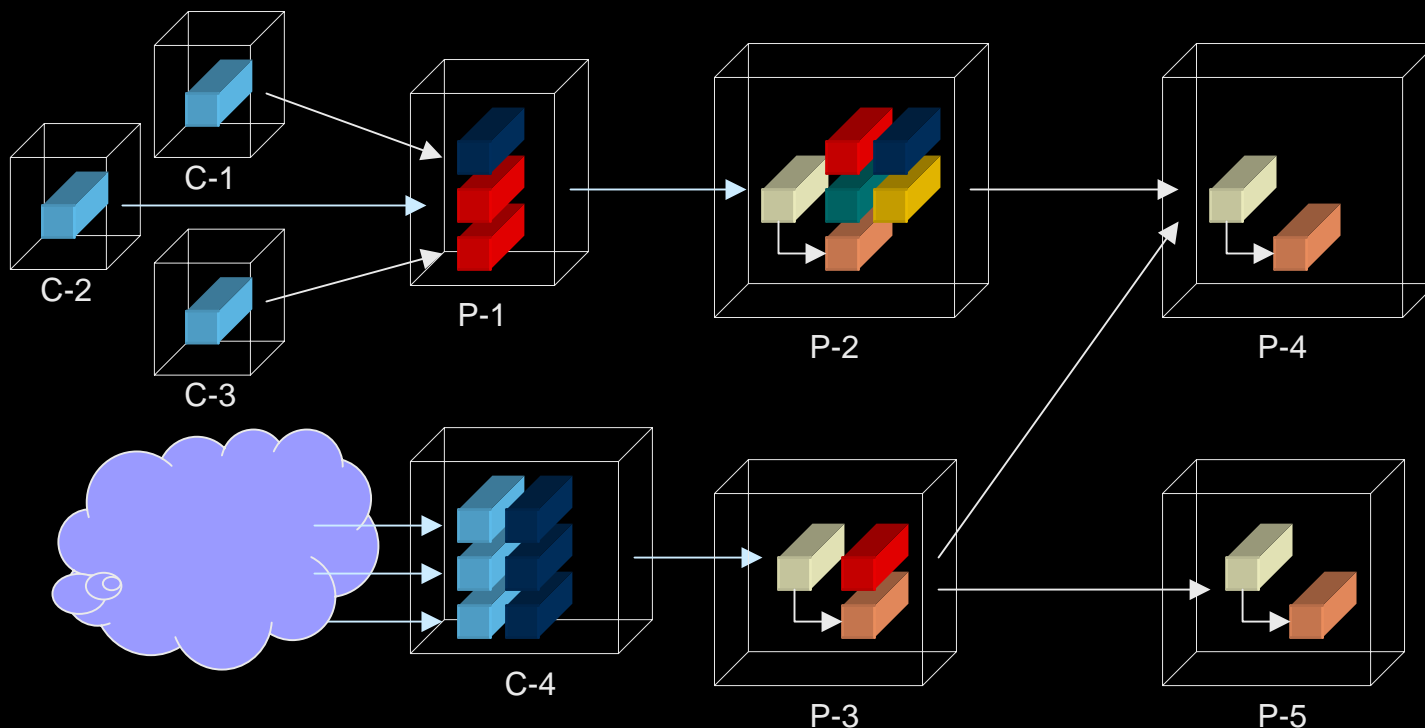
Processing Module

- Dedicated to event management, storage and retrieval
- Hosts can be pipelined to disseminate processing load and enforce privacy policies
- Same data operators as collection module, plus query and archive



Operator Piping

Can be arranged in a variety of ways to accommodate, load, privacy, and functionality considerations



Filter	Tagging	Normalization	Aggregation	Anonimization	DB	Archive

- Problem Space
- Architecture
 - Collection of Events
 - Event Record
 - Management Backplane
 - **Application API**
- Applications
- Status

- Libraries ported to Java, C++, Perl, Shell
- Soap as an interface?
- Basic Operations
 - Query
 - Filter
 - Aggregate
 - Anonymization
 - Trigger
 - Near real-time event feed
 - Control

- Problem Space
- Architecture
 - Collection of Events
 - Event Record
 - Management Backplane
 - Application API
- Applications
- Status

Management Applications

- Common to Both Middleware and Network Applications
 - Comprehensive forensics
 - Performance and QoS
 - SLA verification
 - Use studies
 - Base-lining
 - Historical analysis
 - Notification and alerting
 - Reporting
 - Visualization
 - Resource allocation and billing
- Middleware/General Application Specific
 - Middleware (Shibboleth, P2P, Video, etc.)
 - General (SMTP, Web, DB, DNS, etc.)
 - Discovery
 - Service
 - Policy
- Network Specific
 - Routing behavior and policy verification
 - Connectivity
 - Bandwidth management
 - Discovery
 - Path
 - Flow
 - Device

Security Applications

- Historical event forensics
- Real-time event notification
- Taxonomic risk analysis
- Risk base lining
- Alerts
- Vast array of reports
- Intrusion detection
- Discovery
 - Device
 - Service
 - Policy

- Problem Space
- Architecture
 - Collection of Events
 - Event Record
 - Management Backplane
 - Application API
- Applications
- **Status**

Year One Activity Timeline

Activities	Status	Month				
		Sep – Nov 03	Dec – Feb 04	Mar – May 04	Jun – Aug 04	Sep 04
Pre-Startup	Done	█				
Startup	Done	█				
End-user Requirement Definition/Survey	Active		█			
Architecture and Design			█			
Development, Testing and Distribution				█	█	
Postmortem						█

Major Milestones

- Advisory Group Formed →
- BOF at 12 Oct 03 →
- User Requirements Finalized →
- Architecture/Design Finalized →
- Internal Release →
- Beta Release →
- 0.5 General Release →

Questions?

- For more information,
 - <http://middleware.internet2.edu/e2ed>
 - Or contact me directly, chas@cmu.edu