



While managing business events has long been an important concept when building an enterprise technical architecture, the model as well as the tools that are used to create the architecture have not been able to address the large amount of lower-level data that may or may not have business relevance. Attention to these events and their patterns can provide significant value in addressing operational and business challenges.

With vast additions of data from the IoT, both the challenges and opportunities afforded by this information have exploded and have further underscored the need for a new architectural model.

The **Event Watershed** model provides an approach for propagating, managing and pruning events in the enterprise cloud. In addition it provides seamless access to both real time and historical events which is necessary for purposes of complex event processing, analytics and predictive modeling.

Critical to success of this model is the ability to support the plug and play addition of processing layers and data to the watershed. This is achieved by deploying processing components that support their dynamic addition into a hierarchy and ensuring all data is enhanced to support universal identification and retrieval.

This paper begins by identifying the historical need for an Enterprise Event Bus that can handle the large volume of data represented by many types of lower-level events. (Skip this section if this history is not of interest.)

The paper further describes the structure and key features of the **Streaming Event Watershed** as an approach to achieving a scalable architecture capable of managing vast amounts of data.

MJA Technology LLC's Visual Dataflow Application Builder (VDAB™) provides a reference implementation of this technical architectural pattern based on the visual assembly of processing components. (See <http://vdabtec.com> for details and availability.)

Table of Contents

Evolving Bus Architectures	3
Integration Chaos to a Business Event Bus	3
Point to Point Integration	3
Enterprise Integration	4
Enterprise Integration with Business Process Management	4
Service Orientated Architectures	5
Adding an Enterprise Streaming Event Bus	6
Unstructured Data and Non-Actionable Events	6
Supporting CEP and Analytics on an Event Bus	7
Event Bus - Technical Challenges	7
Introducing the Event Watershed	8
The Event Watershed is Built of Streams	9
Stream Capabilities and Components	9
Sample Stream Relationship	10
Summary Key of Event Concepts	11
Enhanced Event Data and Data Management	12
Enhanced Event Data	12
Event Data Universal Identification	13
Event Propagation Restrictions	14
Event Pruning by Evaporation	15
Event Watershed Data Persistence and Retrieval	16
Hierarchy of Standing Bodies of Water	16
Direct Access to the Appropriate Body of Water	16
Redirected Access to the Appropriate Body of Water	17
Reference Implementation with VDAB [™]	18
Glossary	19

Evolving Bus Architectures

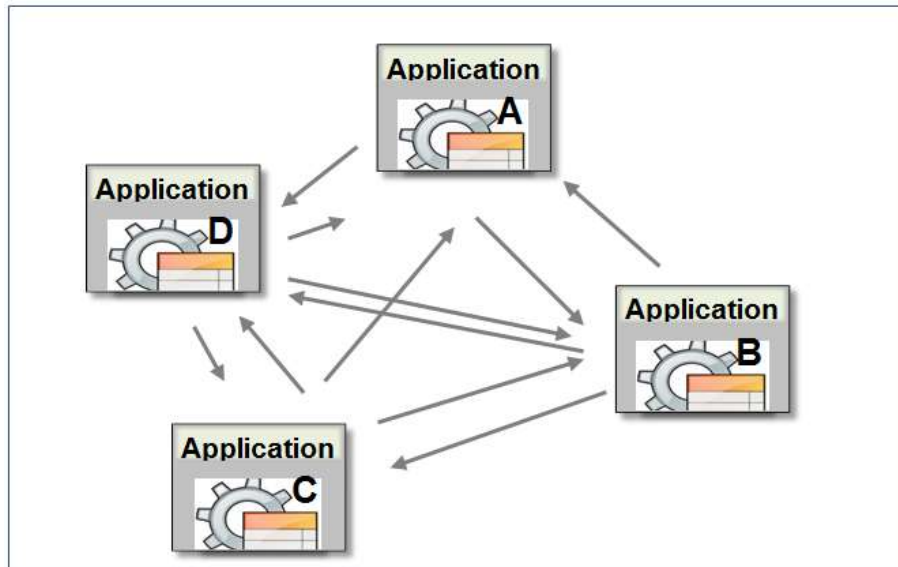
The following is a brief review of how enterprise bus architectures have evolved to address some critical business issues.

Integration Chaos to a Business Event Bus

As soon as industries moved from monolithic mainframes and began to have separate systems with a variety of technologies there became an immediate need to integrate these different systems to each other.

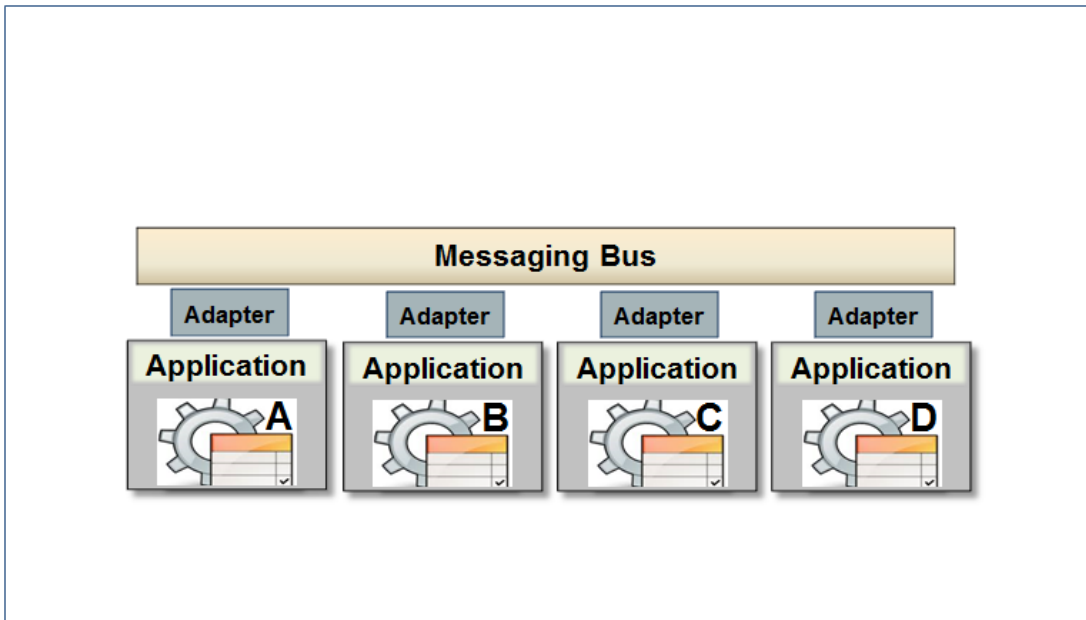
Point to Point Integration

Integrations were put in as needed resulting in a patchwork of often unreliable and difficult to maintain integrations. In many cases each integration became a significant project and there was no guarantee that the integration could be reused when a second application might need the same support. Additionally these integrations often were introduced as departmental projects without any recognition of the difficulty and cost of maintenance.



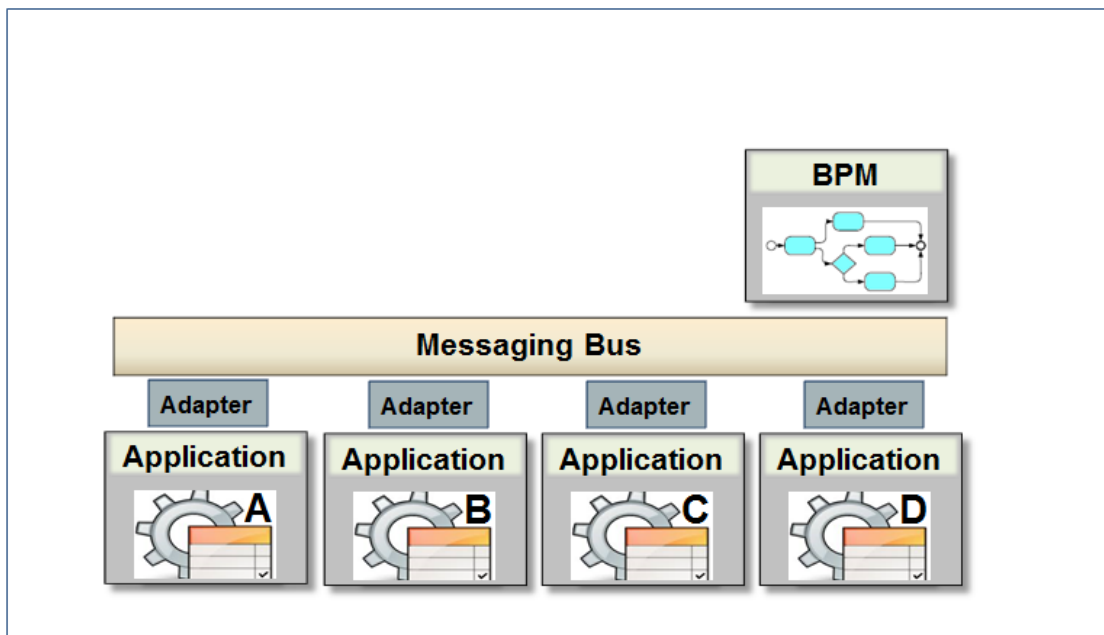
Enterprise Integration

In response to these challenges, enterprise integration tools were developed that used a common messaging bus and adapters or connectors for each application that ensured reliable delivery and opened up the possibility for reuse of these integrations. Standardizing on a tool ensured that management and support could also be standardized.



Enterprise Integration with Business Process Management

Once a reliable messaging bus was in place and application messages were in a well-defined structure, the addition of a BPM engine allowed the enterprise architecture to go beyond just creating orderly integrations and allowed the management of distributed business processes and in some cases new applications using the BPM engine.

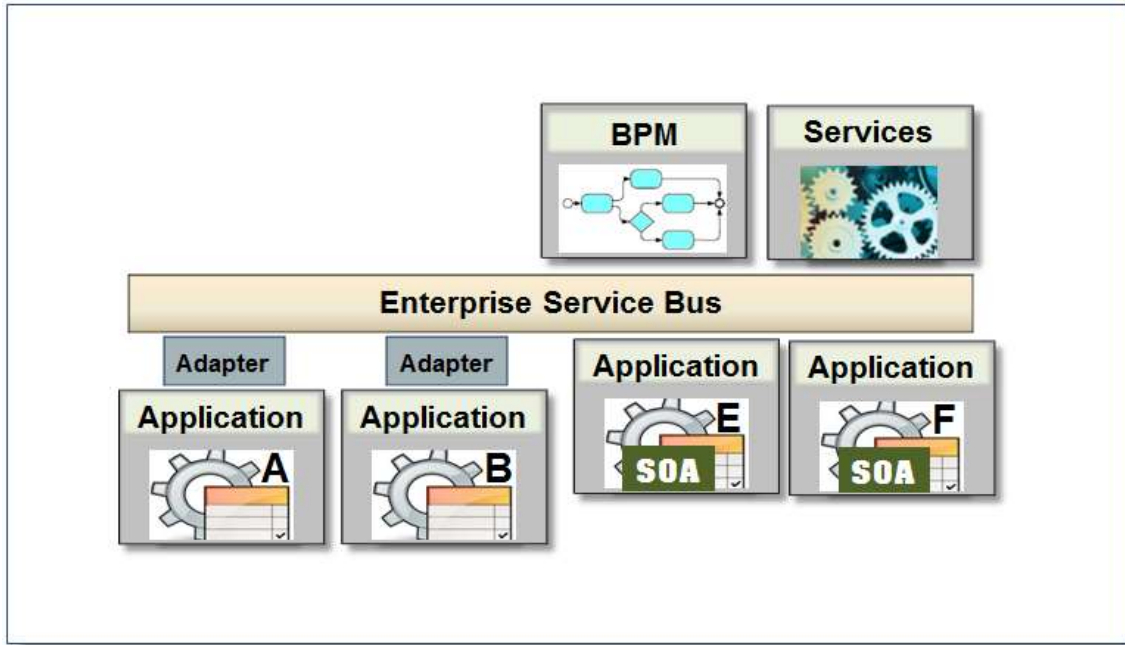


Service Orientated Architectures

The movement toward building applications with separate components exposed as services suggested that in many cases adapters may no longer be necessary and a bus could be built that leveraged service aware applications. The standardization of BPEL and Web services led to the Service Orientated Architecture (SOA) .

While featuring support for synchronous interactions, the bus needed to continue to support asynchronous messaging and “older” applications that required adapters.

Applying the ESB to too many transactions or events often quickly overwhelmed the bus where it could easily become the performance bottleneck for its dependent applications.



While in most case whether or not the ESB is used for specific integrations is made on a project by project basis, an approach to deciding what transactions or queries should have a seat on the overcrowded bus is best based on their enterprise relevance.

Type	Example	On Bus?
Business Transaction	Invoice Generated	Yes
Service Invocation	Return a policy rating for given policy	Yes
System Metric Data	JVM data for a specific app server	No
Data lookup	Lookup an address for customer	Maybe
System Alert	System exceeds utilization	Yes
Log Data	Summarized log data	No
Sensor Data	IoT Sensor Data	No

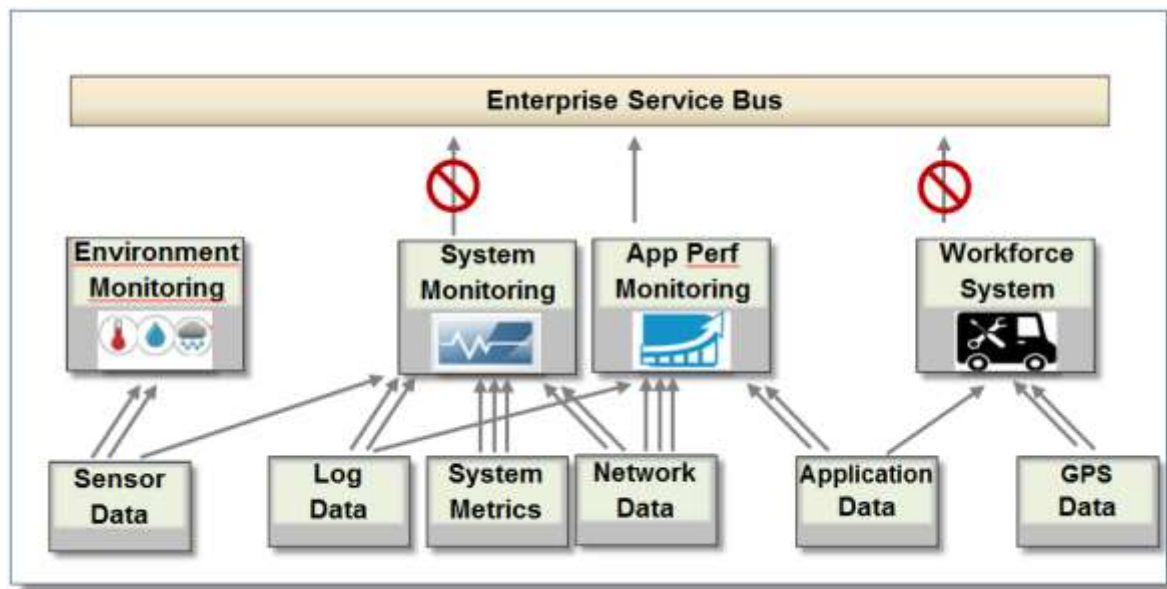
Adding an Enterprise Streaming Event Bus

Unstructured Data and Non-Actionable Events

While events representing actual business events, service requests or other actionable events belong on the ESB, a large amount of unstructured data and other events may be valuable if examined for conditions or patterns.

After examination these events a) may themselves be actionable such as a system condition that requires intervention or b) the patterns may require analysis or intervention.

Currently this wealth of data is generally managed within a specific domain by specific applications. These applications often do not have a mechanism for making use of the ESB when they detect an actionable pattern or conditions.

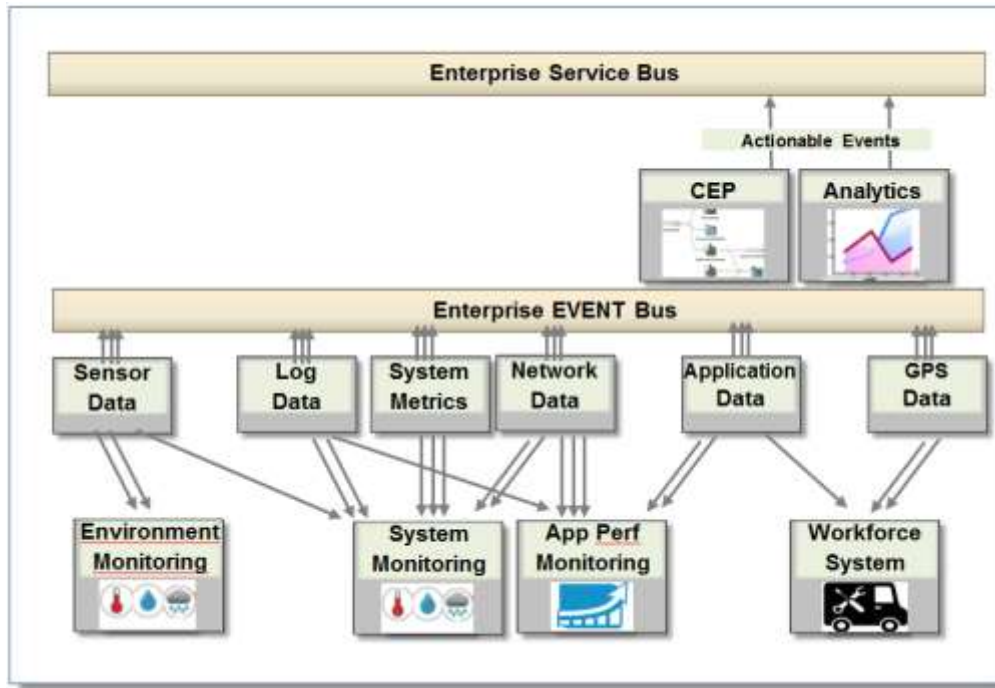


The value of the metric data is limited for a number of different reasons:

- The system that manages it may not be connected to the ESB and therefore all response to the events must be handled within the managing system.
- The event data in these systems is isolated and therefore not available for correlation with other types of metric data.
- The capability to manage certain kinds of data, like log data may be very limited in the available applications

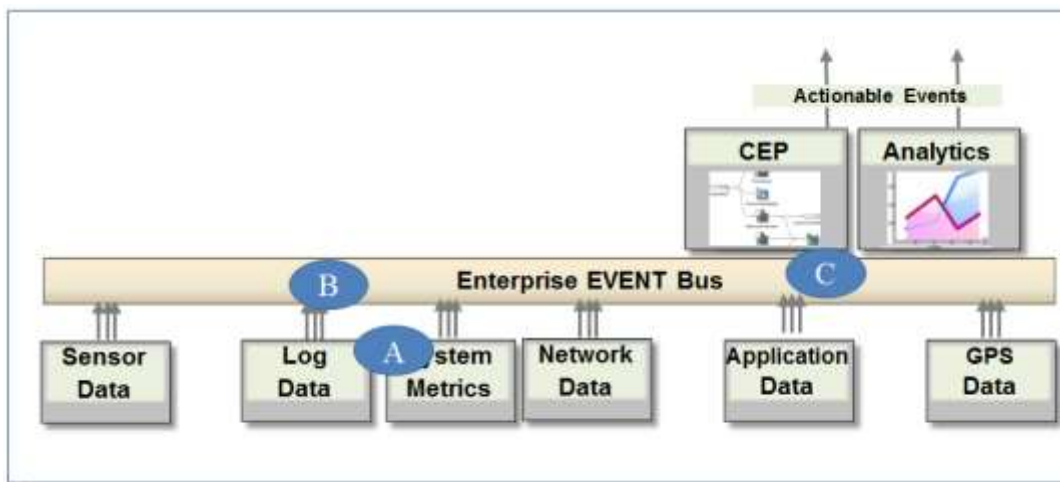
Supporting CEP and Analytics on an Event Bus

Given an appropriate technology that can manage the volume, adding an event bus could provide access to many varied types of lower-level events. Previously contained in a single application where they were difficult to access outside of those applications they can now be exposed to other applications. Complex event conditions across event domains could be analyzed using either a CEP engine or operational analytics and appropriate specific critical conditions or patterns could be identified. Once recognized, actionable events could be generated and responded to by appropriate applications or services



Event Bus - Technical Challenges

While providing a place for valuable lower level events, an Enterprise Event Bus presents significant challenges:



- A. Significant effort would be needed to get a large percentage of the event data on the bus.
- B. The amount of potential traffic on the bus would overwhelm any monolithic centralized bus architecture.
- C. Operational analytics systems would require access to both real time and historical data.

Introducing the Event Watershed

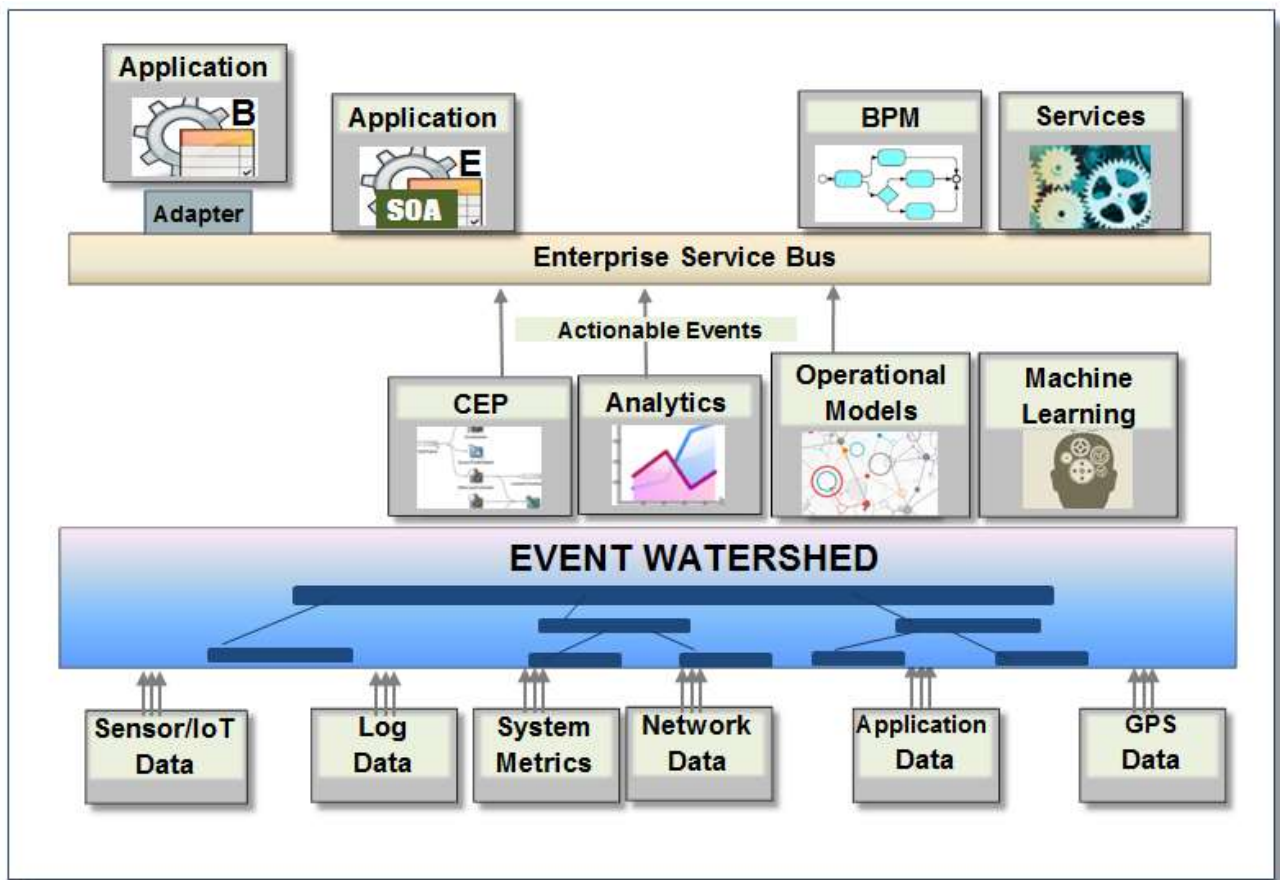
While a single monolithic enterprise event bus cannot manage the volume and access requirements; a multilayered, hierarchical, self-managing component can address this need.

While serving as a real-time bus for event propagation, it additionally has features of an event data store and includes integrated concepts for data pruning, event enrichment and hierarchical data propagation.

Because of Event Watershed **seamlessly handles both real time and historical events**, it is ideally suited to support CEP, predictive modeling, machine learning and operational analytics.

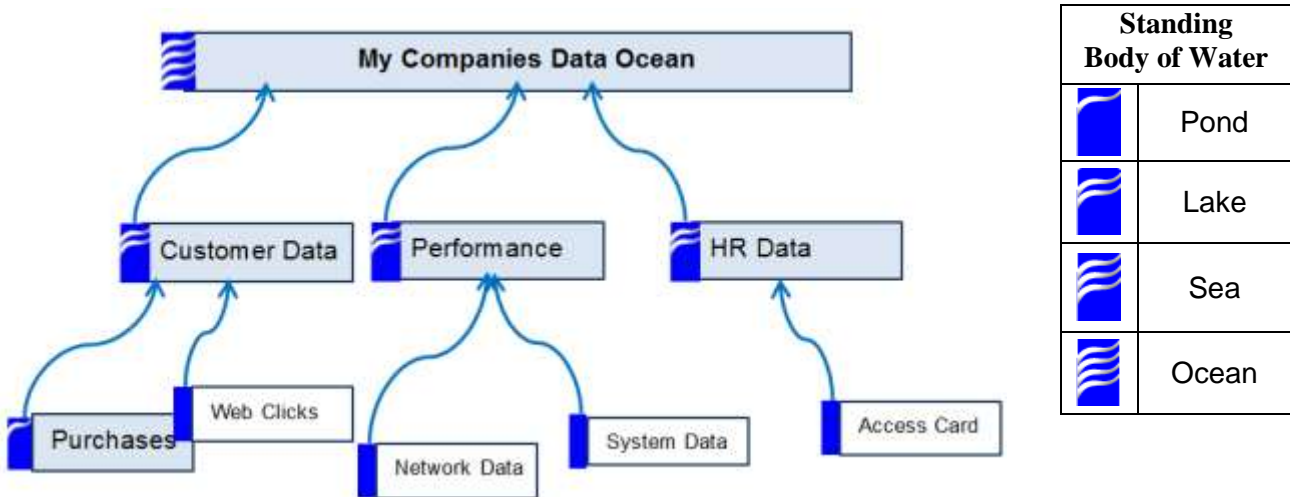
Models developed by analyzing historical data now can now utilize the event watershed to provide responses to modeled conditions and utilize real time data to automatically adjust the models.

The Event Watershed becomes provides an important part of the technical architecture where it is the layer that manages technical events. It combines management components, a unidirectional event bus and event data storage in multiple layers:



The Event Watershed is Built of Streams

The event watershed illustrated as a single row in the technical architecture model is in reality a hierarchy of components each of which can propagate data and persist data. Each of these components is visualized primarily as a stream to move data through the watershed which optionally includes a standing body of water (lake, pond ...) to support persistence of the events moving through the watershed.



The arrangement of these components would be expected to be fairly deep and complex in most organizations, and this complete hierarchy of components is the event watershed.

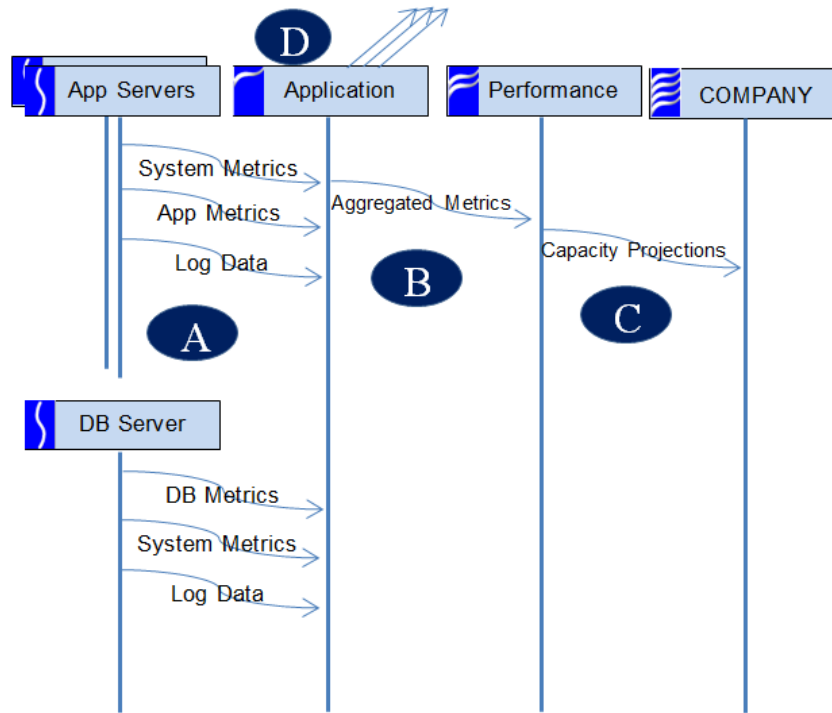
Stream Capabilities and Components

Each of these streams consists of three distinct which have the following key capabilities:

Stream Capabilities and Components			
Propagation	Stream itself		Each stream pushes data downstream
Processing	Flow Container		Each stream can process events in flows
Persistence	Associated Body of Water		Streams can have an associated standing body of water for persistence.

Sample Stream Relationship

The diagram below represents how various stream feed data into each other and how not all events continue to move downstream:



- A. The application server stream and database server stream for a specific application send all relevant application and log data to the stream representing the whole application. The data is persisted at this *Application Stream and Pond* containing and contains all the original data.
- B. Only a limited number of aggregated metrics are sent to the *Performance Stream and Lake* which receives some events from all applications (not shown)
- C. Periodically summary capacity projections are sent to the *Enterprise Ocean*.
- D. Based on the designated relevance time for the original metric events, detailed events may evaporate out of the *Application Pond* when they are no longer considered relevant.

Summary Key of Event Concepts

Key Watershed Concepts	
Events are Immutable	Events are immutable and can be deleted but never updated
Events and Streams	Events can originate or be processed in a stream
Event Enrichment	The headwater or source of the event must enrich events to support complete handling in the watershed
Event Flow Programs	Processing within a stream is defined using an event flow program
Event Sampling	Events can be sampled for additional real time processing at any point where streams merge as they move downstream
Event Query	Streams that are associated with a standing body of water persist events and the can be queried
Event Evaporation	Unless secured by downstream processing, events “evaporate” from the watershed if they have been designated as having a limited relevance time
Universal Identifier	The universal event designation allows retrieval or processing at any standing body of water including ponds, lakes and seas

Enhanced Event Data and Data Management

Enhanced Event Data

All data managed in the Event Watershed is organized as events. This event data structure is a natural fit for nearly all types of data and includes the time and location where the data was created:

Property	Description
Time	The time this event was created
Source	The place this event was created
Type	An event type classification
Data	An data associated with the event

Data management, propagation and retrieval are enhanced by including additional fields:

Property	Description
Time	The time this event was created <ul style="list-style-type: none">• Time bases should be synchronized
Source	The place this event was created <ul style="list-style-type: none">• Source should be universally qualified
Type	An event type classification <ul style="list-style-type: none">• Type metadata should be maintained
Data	Data associated with the event <ul style="list-style-type: none">• Fields and subfields should be labeled in the data or type metadata
Relevance Scope	The anticipated scope of the event
Relevance Time	The time the event is may be relevant
Checksum	Checksum of immutable properties and Data

Event Data Universal Identification

The model assumes that any data (not just event) found in it is universally identified and therefore can be distinguished from any other data in the Watershed.

Property	Description
Time	The time this event was created <ul style="list-style-type: none">Time bases should be synchronized
Source	The place this event was created <ul style="list-style-type: none">Source should be universally qualified
Type	An event type classification <ul style="list-style-type: none">Type metadata should be maintained
Data	Data associated with the event <ul style="list-style-type: none">Fields and subfields should be labeled in the data or type metadata
Relevance Scope	The anticipated scope of the event
Relevance Time	The time the event is may be relevant
Checksum	Checksum of immutable properties and Data

This universal identification would therefore need to ensure it includes all of the following fields resulting in a unique identifier:

Source	Time	Data Field Label
--------	------	------------------

The source would identify a specific system and component on a system. The Watershed would require some conventions or features that would ensure these are not duplicated:

For example the unique identifier for system utilization for the oracle application on a specific server might include the corporate domain and a specifier for the component or application on the server.

Oracle.Server103.companyA.com	10Jan2013_13:04:032	CPU.PercentUtilization
-------------------------------	---------------------	------------------------

Event Propagation Restrictions

While the components of the event watershed are designed to handle huge amounts of event data, the movement of event data toward consolidation, first in pools and then into the sea or ocean can become unmanageable. Two concepts, which are supported by declarations when the event is originated, limit this large amount of data. Event propagation level limits how far downstream and event is moved.



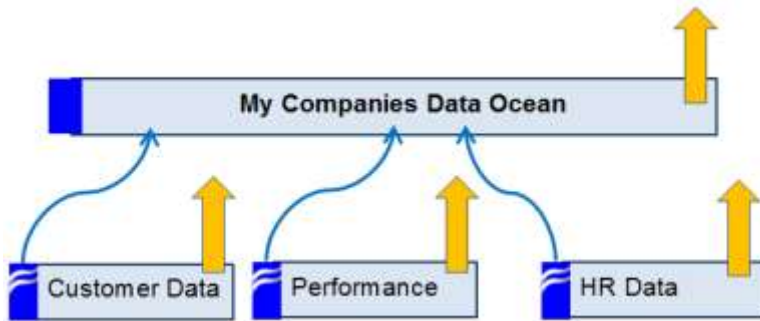
Data movement downstream can be limited by declaring a limited propagation level. Data that is designated to propagate to the **HR Data Lake** will stop there and not propagate to the **Company Ocean**.

The **Relevance Scope** defined in the Enhanced Event Structure is used by each stream processing container to control whether the events it examines are moved further:

Property	Description
Time	The time this event was created <ul style="list-style-type: none"> Time bases should be synchronized
Source	The place this event was created <ul style="list-style-type: none"> Source should be universally qualified
Type	An event type classification <ul style="list-style-type: none"> Type metadata should be maintained
Data	Data associated with the event <ul style="list-style-type: none"> Fields and subfields should be labeled in the data or type metadata
Relevance Scope	The anticipated scope of the event (Stream, Pond, Lake, Sea or Ocean)
Relevance Time	The time the event is may be relevant
Checksum	Checksum of immutable properties and Data

Event Pruning by Evaporation

If nothing is altered downstream, the copies of events that are in a standing body of water will remain there until the relevance time designated by their source or headwater has expired. At this time they evaporate or are pruned from that repository.



When originated, events are assigned a relevance time. Once expired they are purged or evaporate from the data store

The **Relevance Time** is defined in the enhanced data structure.






Property	Description
Time	The time this event was created <ul style="list-style-type: none">Time bases should be synchronized
Source	The place this event was created <ul style="list-style-type: none">Source should be universally qualified
Type	An event type classification <ul style="list-style-type: none">Type metadata should be maintained
Data	Data associated with the event <ul style="list-style-type: none">Fields and subfields should be labeled in the data or type metadata
Relevance Scope	The anticipated scope of the event
Relevance Time	The time the event is may be relevant
Checksum	Checksum of immutable properties and Data

Event Watershed Data Persistence and Retrieval

Data in the entire event watershed needs to be accessible to all the components that sit between the watershed and the Enterprise Service Bus.

Hierarchy of Standing Bodies of Water

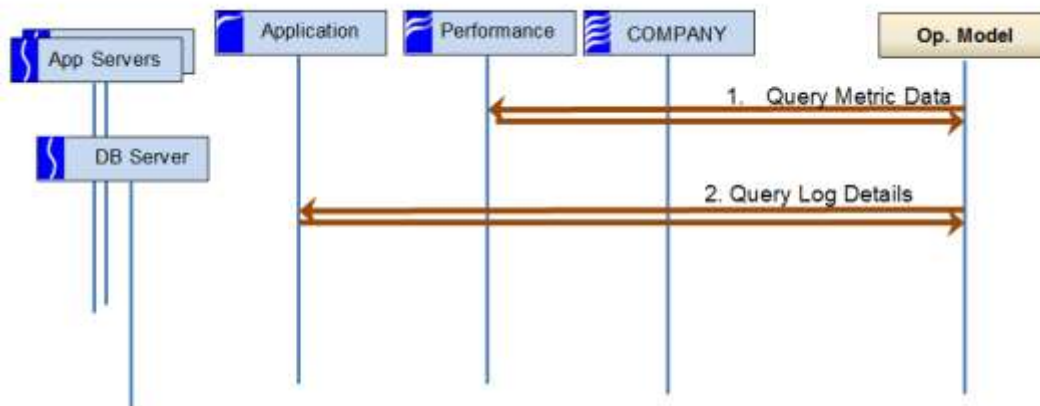
The Standing Bodies of Water in the entire Watershed are hierarchical and the type of body of water should have some business significance. The size of data would dictate the appropriate persistence approach.

Standing Body of Water	Business Significance	Possible Data Size*	Persistence Options
 NONE	There is no standing body of water associated with this stream and as a result data will be propagated but not persisted.	NA	<ul style="list-style-type: none"> • In Memory only
 Pond	A pond would contain an initial persistence point for data which might be considered purely technical with no specific business scope	<2 Gigabytes	<ul style="list-style-type: none"> • local File • NoSQL • RDBMS
 Lake	A lake would contain data that is generally associated with a specific category of data or particular application	2-50 Gigabytes	<ul style="list-style-type: none"> • RDBMS • NoSQL
 Sea	A sea would contain data that is generally associated with a specific group of applications or a department	1-100 Terabytes	<ul style="list-style-type: none"> • NoSQL • HDFS
 Ocean	The ocean would contain data potentially relevant to the Enterprise	100-500 Terabytes	<ul style="list-style-type: none"> • HDFS

* Designating the body of water type has more to do with the business significance than the potential amount of persisted data. As a result these are really just sample persistence sizes. If a stream collected terabytes of network data and was implemented with Hadoop, it might still be designated a pond because it is not expected to be relevant at a known business level. Processing in that stream might aggregate network data summaries which are relevant at a higher level.

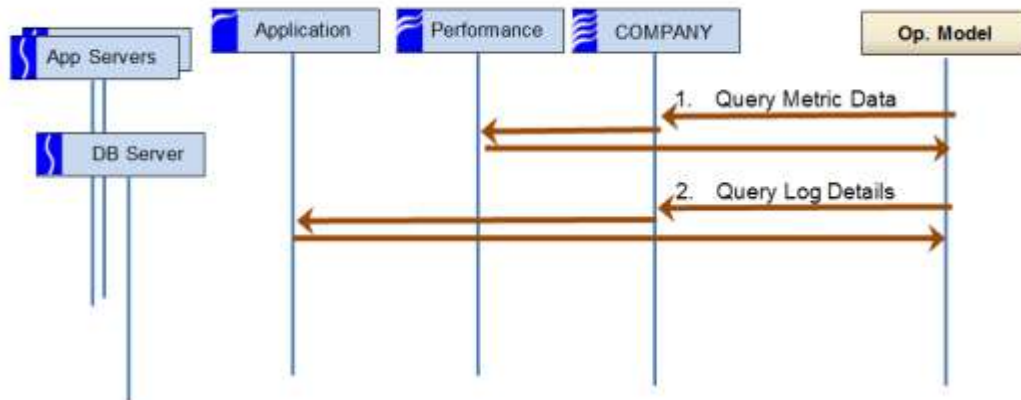
Direct Access to the Appropriate Body of Water

While represented as a single component in the overall Technical Architecture, the Event Watershed is complex and layered. While events may be duplicated in bodies of water downstream, in many cases the event data will not be and must be retrieved from the first body of water. Often an application will know which body of water contains an event or event series. In this case querying can be directed to the appropriate body of water.



Redirected Access to the Appropriate Body of Water

For analysis programs that do not know the location of specific event or event series, they can query any downstream body of water or the ocean and they will be redirected to the proper body of water. Subsequent queries will remember and go directly to the proper body of water:



Reference Implementation with VDAB™

MJA Technology LLC has created the **VDAB™** visual programming platform for building distributed applications and the **VDAB Watershed™** to support the management of enterprise event data with event data flows and hierarchical data propagation. Visit <http://vdabtec.com> for the latest information regarding VDAB.

The **VDAB™** platform which includes the **VDAB Watershed Server** and **VDAB Watershed Client** (Android) are currently concluding early access evaluation and will be generally available in Q3 2017.

The **VDAB Watershed Server** can run on nearly any Unix or Windows systems including very small systems such as the Raspberry Pi's and the Yocto Linux configured for Cisco's IOx initiative. It can use a wide variety of relational databases for storing its enhanced event data.

While VDAB currently supports IoT integration using HTTP and MQTT, additional processing nodes are under development to support integration to a wider variety of smart devices and sensors.

A version of VDAB for managing “lakes” of data is currently under development and works with HDFS and NoSQL databases. This **VDAB Enterprise Server** is expected to be available *in Q2 2018*.

Glossary

Term	Definition
Event	
Actionable Event	An event that is anticipated to require some follow-up activity either by an individual or system.
Business Event	
Technical Event	
Event Series	A sequence of events that represent the same data source and type at different times.
Enhanced Event	An event which contains relevance information for management and a universal event identifier for universal location
Event Stream	An unbounded sequence of events
Event Watershed	A hierarchical tree of event stream components supporting event processing and event persistence
Downstream	A stream toward which other stream data is flowing
Headwater	The stream that originates an event series
Pond	The initial persistence point for events in a Watershed.
Lake	A persistence point for events that have application or departmental significance
Sea	A persistence point for events representing departmental or divisional significance
Ocean	A persistence point for events that have enterprise significance
Evaporation	The automated purging of data from a standing body of water on an Event Watershed